Plataforma de sistema de fitxers distribuït en el cloud

Grau en enginyeria informàtica
Enginyeria de computadors

Director: Juan Jose Martín Pastor
Ponent: Javier Verdú Mulá
Autor: Dídac Plaza Torelló

Resum

Aquest document conté el Treball de Final de Grau, desenvolupat per Dídac Plaza Torelló per a la Facultat d'Informàtica de Barcelona, dintre del grau d'Enginyeria Informàtica amb menció en Enginyeria de Computadors (curs de tardor de 2017-18).

LudiumLab és una empresa que ofereix un servei d'aplicacions en *streaming*. L'aplicació seleccionada per l'usuari s'executa en els servidors de LudiumLab, i s'envia una retransmissió d'aquesta perquè l'usuari pugui interactuar.

Per a les necessitats de LudiumLab, aquest projecte busca trobar una solució òptima, escalable i robusta d'emmagatzemar les aplicacions al núvol. Però en el cas especial de poder ser executades directament al núvol i no només emmagatzemar-les.

Abstract

This document contains the Master Thesis for Degree in Informatics Engineering, developed by Dídac Plaza Torelló, for the Barcelona School of Informatics. This Master Thesis is developed for the bachelor's degree in Informatics Engineering, with a major in Computing (spring course, 2017-18).

LudiumLab offers a service based on streaming. The client selects an application, which is executed on the LudiumLab servers, then a streaming of the application is sent to the client, who can interact with it.

This project aims to find an optimal, scalable and robust solution to store the applications in the cloud. But with a detail, those programs will be executed directly in the cloud.

Agraïments

M'agradaria agrair molt a l'empresa LudiumLab, des del primer moment en què vaig treballar amb ells m'han fet aprendre moltíssim. Gràcies a LudiumLab he pogut dur a terme un treball aprenent de forma pràctica en una empresa, fet que m'ha fet realment adaptar-me a noves tecnologies i adquirir experiència que d'altra forma no hagués sigut possible. Fins ara no conec cap empresa que es preocupi tant dels seus treballadors i aconsegueixi un entorn de treball tan bo amb diferència.

En especial, vull agrair al meu ponent del treball, en Xavi Verdú, qui ha dedicat realment molt de temps en ajudar, fins i tot quan no disposava d'aquest temps! Sense dubte aquest treball no hauria estat possible sense ell.

Aquest treball suposa per a mi el final d'una etapa, l'últim tram del grau en enginyeria informàtica. Per tant el fet de poder treballar en una empresa per dur-lo a terme en aquestes circumstàncies considero que és perfecte, i tothom hauria de fer-ho. Adquireixes experiència professional alhora que fas els últims trams del grau, és ideal com a pont entre la carrera d'estudiant i la professional.

Finalment, cal no oblidar el fet que m'ha fet arribar fins aquí, la meva família i a la meva parella que des de temps enrere sempre m'han motivat a anar a més i més. Sense el suport que m'han ofert, les oportunitats que m'han obert i els ànims que m'han donat no hi seria on sóc ara. Aquest agraïment va per tots vosaltres!

Índex

Il·lustracions del document

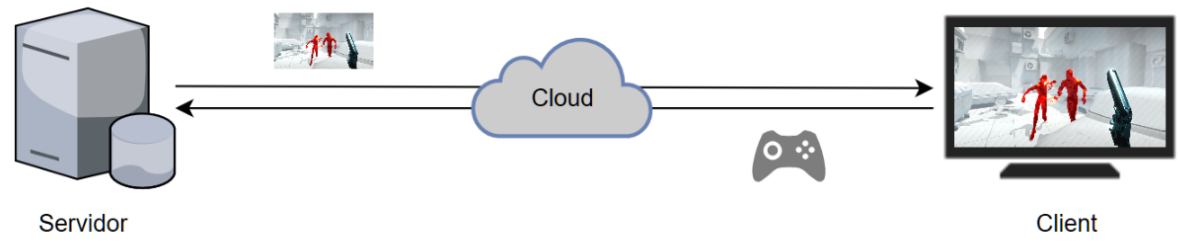
Taules del document

1. Introducció

1.1 Context del projecte

Ja des de fa anys, Internet ha canviat la manera de viure, amb elements nous a causa d'això, com les xarxes socials, i altres antics que s'han modernitzat com els bancs o la manera en la qual comprem. Ara bé, tots aquests serveis necessiten servir el seu contingut l'usuari, de diverses maneres segons el tipus de servei, doncs per exemple no és el mateix un servidor de correu que emmagatzema, envia, i rep correus per a l'usuari, que una aplicació de notícies on no necessàriament cal saber cap dada de l'usuari.

LudiumLab és una empresa que ofereix servei d'aplicacions en *streaming (és a dir, una retransmissió de l'aplicació)*. L'objectiu és poder utilitzar aplicacions des de qualsevol dispositiu. El dispositiu visualitza la retransmissió i permet interactuar amb aquesta, d'aquesta manera l'aplicació no s'executa directament al dispositiu, sinó als servidors de LudiumLab, trencant la barrera dels requisits tècnics. A la il·lustració 1.1 podem veure un exemple, on el servei executa l'aplicació i l'envia al client.



Il·lustració 1.1 Model de funcionament del servei de LudiumLab

A causa del tipus de servei que ofereix ha de servir el contingut als seus servidors perquè executin el producte. Aquest contingut és servit individualment per a cada servidor, això vol dir que cada servidor té una rèplica d'aquest per poder executar el producte. Amb el projecte pretenem preparar una plataforma que substitueixi la rèplica de cada servidor per un sistema més eficient, centralitzat.

Un dels factors més importants d'empreses petites i/o noves que provoca més fallides és el cost de les infraestructures (i en una empresa gran aquest cost es multiplica, fent necessari aplicar alguna solució). Si en comptes d'afegir el cost de llogar un disc dur a cada servidor, creem un disc remot centralitzat per a tots els servidors, l'estalvi en un escenari on tenim centenars de servidors pot ser important. Ara bé, això no només implica una reducció de costos, també implica un canvi en el manteniment, cal tenir en

compte molts més factors, com per exemple l'actualització dels servidors (doncs en comptes d'actualitzar tots els discs, només s'actualitza el disc remot).

1.2 Actors implicats en el projecte

Dintre del projecte, hi ha diverses persones i/o entitats les quals es beneficien i utilitzaran el producte resultant d'aquest, directament i indirectament.

1.2.1 LudiumLab

L'empresa amb la qui es desenvolupa el projecte, LudiumLab, serà la principal beneficiada d'aquest, doncs amb l'objectiu d'aplicar-la dintre del seu negoci pretén obtenir els possibles beneficis d'utilitzar-ho, com per exemple, reduir costos.

1.2.2 Director i ponent

Formant part de l'empresa, el director ajuda en la mesura del possible a què el projecte tiri endavant, obtenint un benefici mutu per a l'empresa i per al desenvolupador del projecte. Juntament amb el ponent s'asseguren que el projecte es desenvolupa correctament i satisfactòriament. Aquests corresponen a Juan Jose Martín Pastor i Javier Verdú Mulá, respectivament.

1.2.3 Desenvolupador del projecte

El desenvolupador del projecte, sent el principal actor del projecte, és l'encarregat de portar-ho endavant, fent la investigació necessària, creant els reports necessaris i gestionant el projecte, validant les seves decisions amb l'ajuda del director i del ponent. També obté el coneixement directament i l'habilitat dins de l'àmbit en el qual es desenvolupa el projecte, essent útil en el futur laboral.

1.2.4 Beneficiaris indirectes

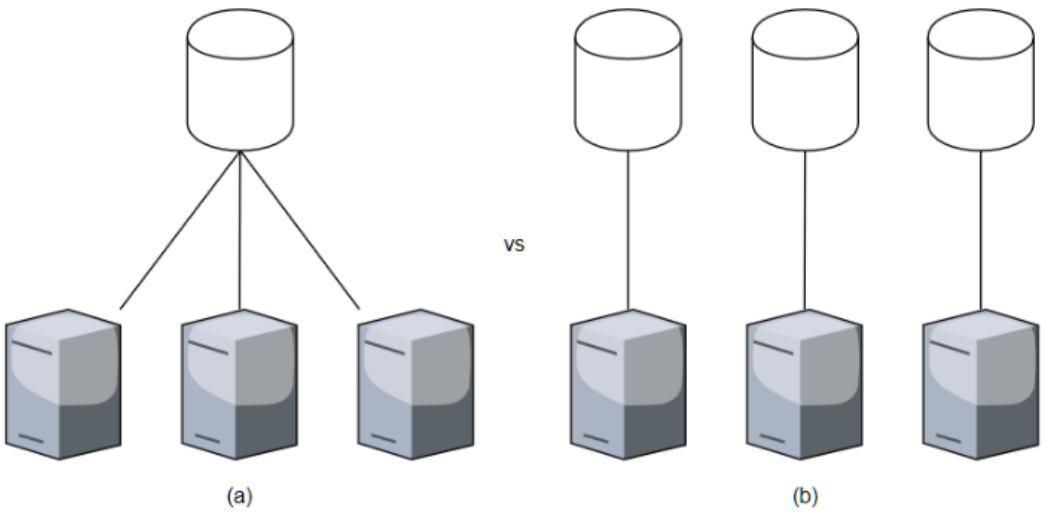
Amb la correcta aplicació del projecte, el producte ofert per LudiumLab millorarà, oferint una millor experiència indirectament als usuaris d'aquesta empresa, així doncs els mateixos usuaris de LudiumLab es beneficiaran indirectament que el projecte tiri endavant i es pugui aplicar correctament.

1.3 Estat de l'art

Emmagatzemar dades en remot pot existir de vàries formes, per exemple servir contingut (sense la necessitat de tindre un sistema de fitxers) sota demanda, com és el cas d'una plataforma que emmagatzemi imatges per una xarxa social. Ara bé, en aquest cas mencionat s'utilitzen Xarxes de Distribució de Contingut (o Content Delivery Network en anglès, CDN per les seves sigles). Una CDN és un conjunt de servidors que

contenen copies d'un o diversos continguts. Aquests servidors estan repartits per diversos punts de la xarxa (usualment el món) per poder oferir el contingut de la manera més eficient possible (seleccionant el servidor més proper per carregar el contingut, per exemple).

Però el cas que necessitem i aprofitarem en aquest projecte **és tindre un sistema de fitxers (com tindríem en un ordinador localment) al núvol en comptes d'un disc per a cada servidor**. D'aquesta forma serem capaços de tindre aplicacions instal·lades i executar-les des de qualsevol dels servidors de la plataforma que ofereix el servei. Podem observar un exemple a la il·lustració 1.2, amb conjunt de servidors connectat a un sistema de fitxers remot ((a), la implementació objectiu) enfront el sistema amb un disc per cada servidor(b), la implementació actual als servidor de LudiumLab.



Il·lustració 1.2 Connexió de servidors a sistemes d'emmagatzemament

Aquest tipus d'accés no es el comú, mentre una CDN pot emmagatzemar el contingut de diverses formes nosaltres necessitem que existeixi una jerarquia de directoris. Altrament les aplicacions no podran trobar les dependències necessàries ni carregar fitxers relatius a la ubicació de l'aplicació.

Dintre de l'àrea d'aquest treball (la connectivitat en xarxa, avaluació de rendiments, comparació de solucions), utilitzarem diversos termes per avaluar el sistema remot. Aquests són els més importants per definir el rendiment del sistema de fitxers remot i entendre el projecte:

- **Latència**: Suma dels retards temporal, expressat en temps. En aquest treball també es té en consideració el retard d'un disc (remot o no) en accedir als fitxers i el retard de la propagació de la informació per la xarxa.

- Throughput: Volum d'informació que pot fluir per la xarxa, en el nostre cas la connexió entre el servidor i el sistema d'emmagatzemament. Expressat en mida partit per temps, com per exemple Bytes/Segon.

- Volum: També conegut com disc lògic, és un emmagatzemament amb sistema de fitxers, accessible a través d'un sistema operatiu. Pot residir en un disc físic, o en xarxa com el nostre cas.

- Client: Servidor de LudiumLab que accedeix al sistema remot per executar aplicacions. Aquest servidor funciona amb sistema operatiu Windows.

Respecte al nostre cas, el sistema de fitxers remot, és molt similar quant a termes utilitzats amb l'àmbit de les configuracions de disc en un sistema local, on diferents discs físics es poden combinar de diverses formes per obtenir millor velocitat o seguretat. Tots els discs junts, independentment de la configuració que tinguin, formen un volum. Per posar-ho en situació en el nostre projecte, el "disc remot" on els servidors es connectaran serà el volum creat per tots els discs que segons el nostre criteri podem organitzar de diverses formes.

Llavors segons el tipus de volum veurem aquests termes, també representats a la Il·lustració 1.3:

- Distribuïts: La informació es distribueix entre els diferents discs que conformen la totalitat del volum, obtenint el màxim espai, però poca seguretat contra fallades. Un exemple de com la informació es distribuïda es pot veure a la il·lustració 1.3, on la informació del volum està copiada a tots els discs (a).

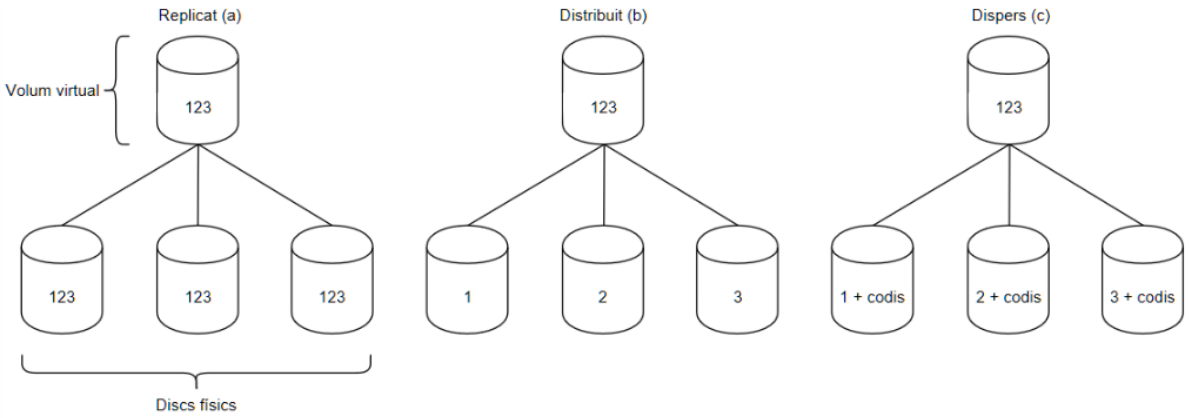
- Replicats: La informació es replica entre els discs, perdent espai total, però en cas que un dels discs falli podem garantir la seguretat de la informació. En determinats casos podem millorar la velocitat de lectura, doncs podem llegir de diversos llocs alhora. Podem veure com funciona a la il·lustració 1.3, a la part (b) observem com el contingut del volum està repartit als discs.

- Dispers: La informació es divideix en trossos que es reparteixen pels discs juntament amb codis d'esborrat, de forma que en cas de fallada es pugui recuperar la informació. Té un límit de discs que poden fallar abans de poder recuperar la informació, que es configura en crear el volum. A la figura 1.3 part (c) podem veure un exemple de com es distribueix la informació.

D'aquests models, els que més veurem seran el distribuït i el replicat especialment. Això es pel fet que el model dispers no es massa utilitzat (en els sistemes remots) doncs

crea càrrega addicional amb els codis de recuperació, augmentant el número d'escriptures necessàries per a cada fitxer. Aquest problema es assumible en certs sistemes físics, o segons el tipus de necessitat, però a nosaltres ens crearia més problemes que els beneficis que ens donaria (com per exemple augmentar la latència).

En aquesta secció presentarem les diferents solucions existents en el mercat.



Il·lustració 1.3 Sistema de fitxers replicat, sistema distribuït i sistema dispers

1.3.1 GlusterFS

GlusterFS [1] és un sistema de fitxers escalable, amb llicència GNU, que ofereix una solució per agregar servidors connectats per xarxa, simulant un sol sistema de fitxers, però darrere està funcionant en paral·lel amb tots els nodes connectats a la xarxa, donant-nos la possibilitat de distribuir la informació de diverses formes, com hem vist a la il·lustració 1.3, distribuïda o replicada per exemple. Els volums creats per GlusterFS s'anomenen bricks en l'entorn de GlusterFS, però el significat seria el mateix que volum, on tots els nodes junts formen un brick.

A diferència de les altres solucions que veurem, no depèn d'un node centralitzat, mantenint-se la xarxa unida per si sola per tots els nodes. Això encara millora si utilitzem el client de GlusterFS, anomenat Gluster Native Client, que permet de forma automatitzada tornar a connectar-se en cas que un dels nodes deixi de funcionar.

És una solució bastant utilitzada, però presenta diversos inconvenients de cara al nostre projecte, doncs està orientada a entorns relacionats amb Linux; I cal recordar el nostre objectiu, que els nodes connectats al volum (clients) siguin sistemes Windows, on el seu client no és compatible. Presenta un bon rendiment i un sistema a prova de fallades ja provat. Actualment és propietat de l'empresa Red Hat Inc.

En el nostre projecte ens centrarem a millorar i superar les restriccions d'aquest sistema, ja que compleix les necessitats de tipus d'accés que necessitem i ofereix un bon rendiment.

1.3.2 HadoopFS (HDFS)

Pensat per al framework Hadoop, HadoopFS [2] és un sistema de fitxers escrit en Java. Orientat a fitxers de gran mida, HDFS divideix en blocs grans els fitxers, i aquests es reparteixen als nodes (DataNodes) connectats al central (NameNode). Aquest NameNode central és l'encarregat de mantenir tot el conjunt de DataNodes (també anomenat clúster) funcionant, replicant la informació necessària, esborrant la informació no necessària, apagant nodes, etc.

A causa del seu origen, està desenvolupat més orientat a aplicacions que utilitzin computació distribuïda, ja que està inspirat en Map/Reduce [3]. Actualment Hadoop és bastant utilitzat en multitud d'empreses i projectes [4], on HadoopFS és molt útil.

Encara que és una solució utilitzada dintre de l'àmbit, per a les característiques del nostre projecte no és el més adequat perquè no és un dels tipus de software mencionats anteriorment.

1.3.3 Ceph

Ceph [5] és una plataforma per emmagatzemar informació, que busca la fiabilitat replicant la informació i distribuint-la en els diferents nodes, tenint com a element diferenciador el fet que Ceph emmagatzema la seva informació en el que anomenen objectes, que es reparteixen pels nodes. L'emmagatzemament en objectes és diferent del comú, no es basa en una jerarquia per organitzar la informació, sent molt eficient per volums de dades molt grans [6].

Ceph és format per 3 principals components:

- Nodes monitors: Contenen l'estat dels altres nodes i els monitoren.
- Nodes de metadades: Contenen la informació relativa a la informació emmagatzemada pel mateix sistema, com el directori on és o el seu inode (informació relativa als atributs de la informació i la seva ubicació dins dels discs).
- Nodes d'emmagatzemament: Contenen la informació pròpiament, dins del sistema on són muntats els nodes.

Usualment és comparat amb GlusterFS, ja que els dos són sistemes descentralitzats (no pengen d'un únic node), són ideats per ser a prova de fallades, tenen una gran disponibilitat (al replicar la informació) i no depenen de cap hardware concret. En el nostre cas com veurem en el següent punt, no ho compararem nosaltres degut a interessos de l'empresa.

1.3.4 EFS

Elastic File System [7] és un servei que ofereix Amazon Web Services (AWS)[1], on només pagues pel que utilitzes. AWS ofereix una infraestructura que actua com a disc, i et connectes a l'adreça que t'ofereixen. Com a avantatge, no has de preocupar-te de la mida del disc (aquesta creix segons emmagatzemes coses, no té límit teòric més que el que ets disposat a pagar), ni que l'ordinador on és falli, doncs de tot això s'encarrega AWS (qui s'assegura de garantir-te una bona fiabilitat).

Com a contra per al nostre sistema tenim que funciona **només** amb NFS versió 4 (sigles de Network File System, un protocol d'accés a sistemes en xarxa), Windows només permet connectar-se com a client amb la versió 3 de forma nativa. El fet de dependre d'AWS implica que si volem migrar a un altre proveïdor haurem de buscar una altra solució.

Una característica important és que la velocitat que ens ofereix AWS sobre aquest disc depèn directament de quant espai estem utilitzant. Per cada Terabyte emmagatzemat disposem de 100 Megabytes per segon durant 12 hores al dia, que també es pot entendre com 50 Megabytes per segon de forma continuada [8]. El límit màxim depèn de la zona, encara que es pot ampliar contactant amb AWS.

En el nostre cas, compararem aquest servei amb GlusterFS, degut a interessos de l'empresa, juntament amb GlusterFS per al seu reduït cost de desplegament i facilitat d'integració amb el servei de LudiumLab.

1.3.5 Altres agents

Dintre d'aquest àmbit, diverses empreses també ofereixen sistemes d'emmagatzematge remot, com el cas d'AWS amb el seu servei S3, són sistemes pensats per emmagatzemar fitxers multimèdia o contingut. Pel nostre projecte no ens interessen tant com les solucions anteriors perquè no estan ideades per treballar ni com a disc en un sistema operatiu, ni com un sistema de fitxers jeràrquic.

[1] https://aws.amazon.com

1.3.6 Sistema propi

Entre altres opcions, en un projecte d'una empresa entra la possibilitat de crear un sistema de fitxers propi. Crear un sistema de fitxers propi implica una inversió molt més gran. Especialment en una empresa petita i amb el temps disponible, no es viable. És molt més viable el fet d'adaptar una de les ja mencionades.

2. Abast del projecte

2.1 Formulació del problema

LudiumLab actualment no utilitza un sistema de fitxers distribuïts com es pot observar a la il·lustració 1.1, on la connexió entre el servidor i els fitxers és local, i no per xarxa. LudiumLab necessita estudiar i provar quina es la solució més adequada, ja que les aplicacions que executa requereixen un throughput semblant al d'un disc físic per oferir una experiència satisfactòria.

L'empresa té servidors que executen la seva plataforma, però tots ells tenen la seva informació de forma local a cada servidor Windows. Un gran número de vegades, aquesta informació de cada servidor és la mateixa, de forma que es podria aplicar un mètode molt més eficient, en comptes de llogar un disc dur per a cada servidor, ubicar la informació en un sistema de fitxers remot, estalviant diners i fent-lo més eficient. Això també implicaria altres avantatges, com el fet de gestionar un sistema de fitxers en comptes de centenars cada vegada que s'actualitza el producte (en un entorn de test fer canvis a un sol servidor és factible, però quan tens un número molt elevat de servidors no és viable aplicar a cadascun el canvi, la feina pot durar mesos, inclòs més si es tenen centenars o milers d'ells). El fet de tindre un sistema centralitzat implicaria un sistema més eficient, més barat, i directament una gestió més simple del contingut.

LudiumLab vol aplicar una de les solucions mencionades anteriorment, però té unes necessitats diferents de la norma, o fora del que podria ser en la majoria de les empreses, on per exemple l'execució dels servidors és en Windows, i la gran majoria d'optimitzacions i eines per als servidors estan més orientades a distribucions Linux.

A més, hem de garantir la disponibilitat del sistema de fitxers, si un node cau, no podem permetre que tots els servidors es quedin sense sistema de fitxers. Aquesta funcionalitat està ja implementada en solucions com GlusterFS, però només per al seu client natiu, no disponible a Windows.

2.2 Objectius

L'objectiu del projecte és desplegar i preparar un servei centralitzat de disc remot capaç d'emmagatzemar aplicacions, on ordinadors amb sistema operatiu Windows puguin connectar-se i executar-les de forma eficient. Una vegada desplegat i integrat comparar amb una altra possible opció per avaluar els rendiments.

Passos essencials a seguir:

1. Estudiar les configuracions més adequades per al sistema de fitxers cloud.
2. Familiaritzar-se amb el sistema de fitxers escollit.
3. Avaluar el hardware adequat per obtenir un rendiment eficient amb la configuració escollida.
4. Desplegar una primera plataforma i avaluar rendiment.
5. Instal·lar aplicacions i executar-les en remot.
6. Modificar plataforma per configurar connexió contra caigudes.
7. Escollir una segona opció, i familiaritzar-se amb ella.
8. Desplegar la segona opció i comparar el rendiment.

2.3 Abast del projecte i possibles obstacles

El fet de substituir els discs físics dels servidors per la connexió a un sistema de fitxers remot compromet a què el sistema de fitxers ha de tindre un rendiment semblant o proper al sistema de fitxers local. Si un servidor vol executar una aplicació, no es viable que la latència sigui excessivament gran.

Cal buscar una solució que satisfaci les necessitats de LudiumLab, i un d'ells és el fet que el canvi sigui el més transparent possible al rendiment dels servidors Windows. Llavors si resumim el nostre problema a solucionar, obtenim els següents requisits del projecte:

- S'ha de garantir un throughput i latència similars als discs físics dels servidors.
- El sistema de fitxers ha de permetre que les aplicacions puguin ser executades més d'una vegada alhora per diferents clients connectats al sistema de fitxers remot.
- La connexió al sistema de fitxers remot ha de ser a prova de fallades, si un node dels que compon el volum remot cau, la connexió ha de ser capaç de mantenir-se utilitzant els altres nodes disponibles.

3. Metodologia a seguir

3.1 Scrum

Dintre de LudiumLab la metodologia utilitzada en l'empresa és Scrum [9], una metodologia àgil que divideix les tasques principals en *sprints*, permetent planificar el temps que duraran, i subdividir-les en tasques diàries, permetent la realització de diferents tasques diàries.

Aquesta metodologia és adequada per aquest projecte, perquè no podem assegurar quins problemes es presentaran (problemes del mateix desenvolupament, no del projecte). Utilitzar Scrum ens donarà flexibilitat en cas d'haver d'adaptar el projecte a possibles canvis o diferents millores.

Encara que és una metodologia més aplicada a treballar en equip, utilitzar-la ens facilitarà portar al dia el projecte amb l'equip o el director, guiant-la de manera més fàcil cap al camí correcte, dividir el projecte en etapes ben indicades i dur a terme feina més clarament dia a dia.

3.2 Eines de seguiment

Dins de LudiumLab s'utilitza Jira [10], una eina que permet aplicar diverses metodologies per fer seguiment dels projectes, entre elles Scrum, registrant totes les tasques fetes, el temps emprat, fer seguiment dels bugs i problemes, i enregistrant els sprints del projecte.

A part del seguiment, dintre de Jira podem gestionar el projecte i guardar arxius relacionats amb el projecte entre d'altres. En ser un software per a empreses, tenim l'avantatge que el projecte pot ser observat pels superiors de l'empresa, facilitant la comunicació i compartició d'informació. A més dintre de Jira existeix la integració amb un altre software, Confluence [11], una eina per crear reports dels projectes.

Jira i Confluence són unes eines de pagament, de la qual LudiumLab en té llicencia per utilitzar-la, propietàries d'Atlassian.

3.3 Metodologia de treball

Dintre de LudiumLab, a causa del tipus de feina, moltes vegades el desenvolupament es pot dur a terme des de qualsevol entorn, no necessàriament les oficines de LudiumLab. Principalment les eines utilitzades per al projecte seran servidors remots per fer les proves i la implementació del sistema.

3.4 Validació del projecte

Seguint la metodologia Scrum, hi haurà reunions contínues amb els col·laboradors del projecte, amb els quals s'anirà avaluant l'evolució del projecte, juntament amb la contínua implementació dels canvis, per als quals s'hauran de crear tests per comprovar el correcte funcionament del projecte. Dintre del projecte, es comprovarà de forma continuada els resultats obtinguts de diverses solucions, per assegurar el millor rendiment possible i assolir els nostres objectius.

Per tal de garantir que la solució que es desenvolupa sigui adequada, durant els diferents punts i avenços del projecte es comprovaran vàries coses:

- Instal·lació i execució d'aplicacions sense errors.
- L'execució d'aplicacions es fa a una velocitat adequada per ser utilitzada.
- Se simularà la caiguda del sistema per testejar si el sistema d'alta disponibilitat segueix funcionant.
- Anàlisi de la velocitat del sistema, no de l'execució d'aplicacions, sinó del throughput que es pot extreure del sistema de fitxers.

4. Planificació

4.1 Especificació de les tasques

A continuació detallarem les tasques, junt amb el seu conjunt de dependències i subtasques. Aquestes tasques són descrites en ordre cronològic.

4.1.1 Gestió de projectes (GEP)

Aquesta tasca engloba tots els lliuraments corresponents a l'assignatura de GEP. Dintre de l'assignatura hi ha set lliuraments, dels quals el sisè i el setè s'entreguen el mateix dia, i van junts pel fet que el setè és la presentació corresponent al sisè lliurament. El total de subtasques suma 75 hores.

4.1.2 Familiarització amb GlusterFS

En aquest apartat, s'engloba tota la part de lectura, aprenentatge i prova de desplegament de GlusterFS. Així doncs podem identificar dues grans subtasques:

- Recerca i documentació
- Desplegament de GlusterFS

Entre elles dues hi ha una dependència directa, perquè fan falta els coneixements sobre GlusterFS per poder desplegar la plataforma. Dintre de la tasca de desplegament també s'inclou el possible temps que ocasionin problemes i errors durant el procés, el cost estimat és de 65 hores, però entrellaçades amb la primera tasca (Gestió de projectes).

4.1.3 Testeig de GlusterFS

Una vegada ja tinguem la plataforma funcionant, dintre d'aquesta tasca es connectarà amb els servidors per comprovar la correcta connexió, una vegada comprovat que funciona, s'avaluarà el rendiment obtingut com a sistema de fitxers remot, i segons el rendiment obtingut, es podrà escollir quin és el hardware adient per no sobrepassar costos ni quedar-nos curts quant a requeriments del projecte.
Llavors identifiquem tres subtasques:

- Test de la plataforma com a disc als servidors
- Anàlisi del rendiment
- Selecció del hardware adient

Aquesta tasca té una relació directa amb la tasca dos, perquè sense el desplegament de GlusterFS no podem fer cap de les subtasques (que alhora internament depenen

entre elles, en l'ordre que apareixen, respectivament). Aquesta tasca serà en una finestra temporal més petita que l'anterior tan superposada amb GEP (primera tasca), encara que comportarà la mateixa feina. El cost aproximat es preveu en 65 hores.

4.1.4 Alta disponibilitat

En aquesta tasca s'investigaran les possibles solucions que es poden implementar per aconseguir l'Alta Disponibilitat, desplegar-les i provar-les. Doncs entenem dues tasques principals:

- Recerca sobre les solucions
- Desplegar i configurar la solució

Entre elles tenen una relació directa, al ser el mateix cas que la tasca dos, recerca més desplegament, a més de dependre directament de les anteriors (sobretot la tasca dos) al necessitar la plataforma per testejar-ho. Però la informació respecte al nostre cas és molt més limitada, considerant necessari més temps per a la finalització de la tasca, en aquest cas, 85 hores.

4.1.5 Familiarització amb EFS

A causa de ser un sistema de fitxers gestionat per AWS, la part de desplegament es més simple. Per tant amb aquesta tasca es pot englobar no només la familiarització, sinó també el desplegament i l'avaluació de rendiment.

Dintre d'aquesta tasca identifiquem aquestes tasques:

- Recerca i documentació
- Desplegament d'EFS
- Testeig de la plataforma amb EFS
- Avaluació de rendiment

En tindre ja coneixements de sistemes i ser un sistema semblant, al conjunt de tasques s'assignen 75 hores. Entre les subtasques hi ha relació directa, però la tasca global no en té amb les anteriors.

4.1.6 Documentació i presentació

La tasca final correspon a la finalització de la documentació i a preparar la presentació per a lectura del TFG, ja que aquestes dues subtasques representaran les dues setmanes finals, unes 50 hores aproximades. Aquesta tasca depèn directament de totes les anteriors per a podes omplir de contingut el document i la presentació.

4.2 Estimació de temps per tasca

Com a mesura de les tasques descrites en el punt anterior, a la Taula 4.1 podem veure resumit les hores previstes de cada tasca i el conjunt total estimat del temps necessari per dur a terme el projecte:

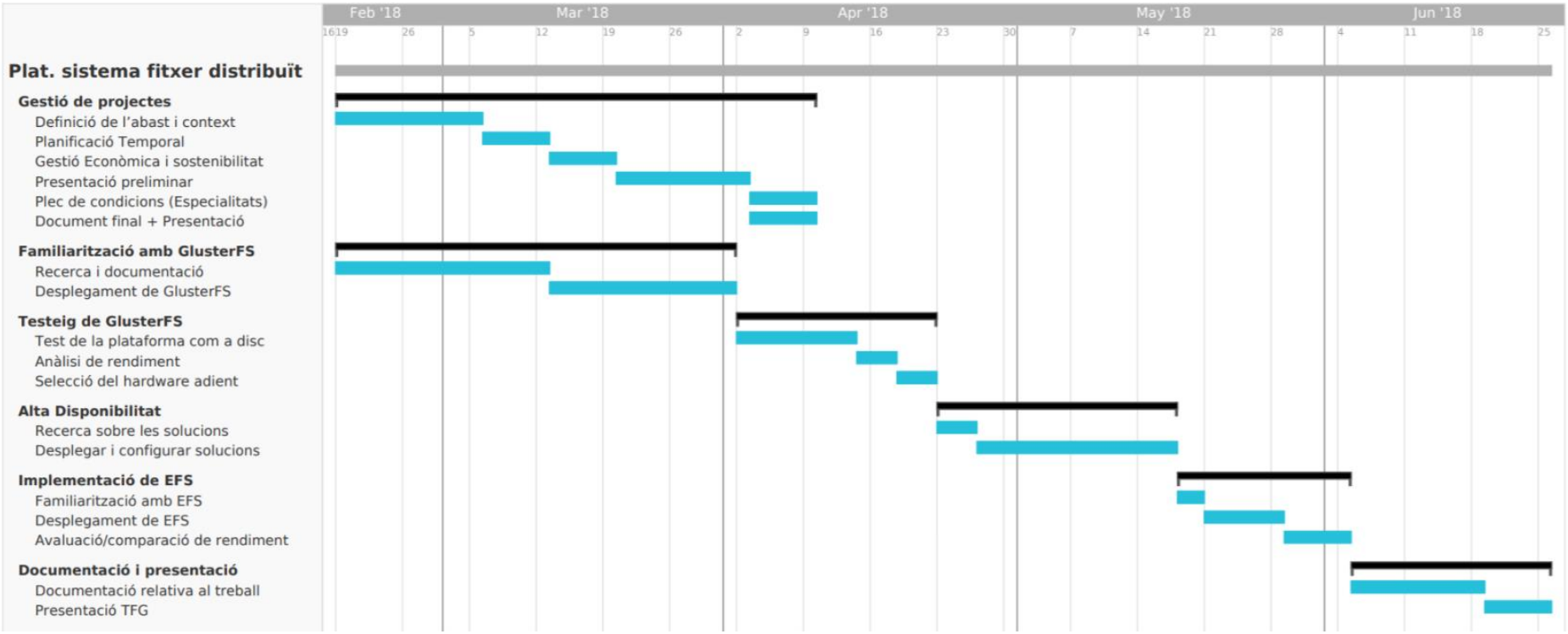
Tasca	Hores previstes
Gestió de Projectes	75
Familiarització amb GlusterFS	65
Testeig	65
Alta Disponibilitat	85
Implementació en EFS	75
Documentació i presentació	50
Total	415

Taula 4.1 Hores previstes

4.3 Diagrama de Gantt

Els diagrames de Gantt són una eina per planificar tasques durant un període de temps. En el nostre cas ens serveix per visualitzar aproximadament com queden repartides les tasques al llarg del termini del projecte.

En el diagrama de Gantt de la il·lustració 4.1 podem veure les tasques mencionades a la Taula 4.1:



Il·lustració 4.1 Diagrama de Gantt

4.4 Recursos

Per al desenvolupament del projecte s'utilitzaran diferents recursos, hardware, software i personal, els quals ens permetran fer viable el projecte i/o facilitar-ho.

4.4.1 Recursos hardware

- Servidors remots, on les seves característiques oscil·laran segons l'estudiant consideri que són els més adequats rendiment/preu per a executar el software necessari per al projecte, i el número d'aquests segons el tipus de sistema de fitxers remot que es vulgui implementar.

- Ordinador portàtil Dell XPS 15 amb connexió a internet, amb el qual es connectarà als servidors remots per configurar-los i preparar-los per a complir la funció del projecte.

- Càmera per a la gravació del lliurament en vídeo corresponent a l'assignatura GEP.

4.4.2 Recursos software

- Putty, software per a la connexió mitjançant SSH als servidors remots [22].

- TightVNC, software d'escriptori remot, mitjançant el protocol VNC [23].

- ATTO Benchmark, software per extreure informació relativa al rendiment dels discs, velocitat, operacions per segon, etc [24].

- GlusterFS, software capaç d'agregar discs per crear un sistema de fitxers distribuït [25].

- Jira, software per al seguiment de tasques i bugs [10].

- TeamGantt.com, web per a la creació de diagrames de Gantt [26].

- Google Drive, servei per emmagatzemar arxius i editar documents [27].

4.4.3 Recursos humans

En el cas d'aquest treball final de grau el desenvolupament correspon únicament a l'estudiant que treballa a temps complet, 40 hores setmanals, a l'empresa LudiumLab (amb la qual desenvolupa el treball). La dedicació al desenvolupament del treball serà

gran part d'aquestes hores, depenent de les altres tasques que apareguin setmanalment relacionades amb el lloc de treball que ocupa l'estudiant.

4.5 Desviacions i pla d'actuació

La probabilitat que alguna tasca no s'executi en el temps establert és present en el treball, dintre de les hores planejades per a cada tasca ja es contempla la possibilitat que hagin de solucionar-se errors.

En cas que una tasca es finalitzi abans del planejat, es comença la següent, no presenta cap inconvenient. En cas que una tasca tardi més del previst se seguirà amb ella, si el temps dedicat puja massa, es prioritzaran primer el desplegament dels dos sistemes diferents (GlusterFS i EFS) per a poder realitzar la tasca principal del treball, el desplegament d'una plataforma en el cloud, i la comparació.

Per seguir un correcte fil de treball, setmanalment es comprovarà l'avenç del projecte juntament amb el ponent i el director del treball. Així doncs en cas estrictament necessari es pot prescindir (perdent funcionalitats) d'alguns punts per complir l'objectiu bàsic, assegurant així la finalització del treball.

4.5.1 Desviacions durant el projecte

Una vegada avançat el projecte, no hi ha hagut grans canvis, l'únic aspecte important és el canvi de CEPH a Amazon Web Services EFS. A causa de trobar certs aspectes que el fan més atractius que CEPH per a comparar-lo, a l'oferir la mateixa solució però amb una infraestructura diferent dona un plantejament nou, i la comparació no és simplement de velocitat o cost, també de manteniment, escalabilitat i estabilitat dels dos tipus d'infraestructures.

Això implica canviar el punt respecte a muntar el sistema en CEPH per comparar, a passar a desplegar-ho amb EFS. El cost d'EFS serà mínim, perquè es paga per l'ús i només farem una comparativa. Igualment aquest canvi és contemplat al pressupost i al diagrama de Gantt.

Després de diversos mesos, al punt de la segona fita del treball tot era com era planificat, el sistema amb EFS estava funcionant, i en procés la comparativa i l'anàlisi de les plataformes.

Fins al moment, la metodologia de treball que es va decidir al principi ha funcionat força bé, arribant sense problema a les dates planificades. Així doncs la metodologia original es manté.

5. Gestió econòmica i sostenibilitat

5.1 Pressupost

Seguint la planificació del punt anterior, fem una estimació del cost del projecte, on considerarem costos hardware, software i humans, directes i indirectes.

5.1.1 Costos hardware

En aquest apartat detallem els costos hardware, els possibles riscos i les amortitzacions corresponents. En el nostre projecte, els servidors es lloguen per hores, llavors no tenen en compte les amortitzacions al no comprar-se inicialment.
El càlcul de les hores de servidors correspon a la part de planificació en la qual s'utilitzaran servidors, multiplicada per tres (utilitzarem tres servidors).
*El risc està basat en la fiabilitat de DELL de 3.42% segons un estudi de tecnicosclic.com [12]. En la següent taula 5.1 podem veure tot el mencionat:

Producte	Preu	Unitats	Vida útil	Estimació
Dell XPS 15	1900 €	1	5	89.60 €
1 hora de servidor remot	0.0102468182 €	604	-	6.19 €
Càmera	369 €	1	4	0.33 €
Risc*	64.98€	-	-	64.98 €
			Total	161.10 €

Taula 5.1 Costos Hardware

5.1.2 Costos software

En l'apartat software tindrem en compte tots els costos relacionats amb el programari utilitzat per dur a terme el treball, encara que quasi tots són programaris lliures sense cost, el sistema EFS té un cost de 0€ al no excedir els 5GB mensuals que ofereix Amazon de forma gratuïta. Caldria destacar el fet que la llicència de Jira és per a 10 usuaris, durant 6 mesos (880 hores), però és un cost indirecte que ja era a l'empresa, igualment tingut en compte a la Taula 5.2.

Producte	Preu	Unitats	Estimació
Putty	0 €	1	0 €
TightVNC	0 €	1	0 €
ATTO Benchmark	0 €	1	0 €
GlusterFS	0 €	1	0 €
EFS*	0.25 €/GB	<1 GB	0 €
Jira	8.13 €	1	0.38 €
TeamGantt	0 € (Versió gratuïta)	1	0 €
		Total	0.38€

Taula 5.2 Costos Software

5.1.3 Costos humans

En aquest cas, el projecte és dut a terme per a LudiumLab mentre l'estudiant treballa a jornada completa amb ells. L'estudiant treballa per a LudiumLab com a administrador de sistemes, llavors el preu real del projecte es el que paga LudiumLab per l'estudiant com a treballador en qualitat d'administrador de sistemes, a 16 € l'hora.

Es dividirà per les etapes del projecte per a una millor visualització del cost, com podem observar a la taula 5.3:

Producte	Preu/Hora	Hores	Estimació
Gestió de Projectes	16	75	1200 €
Familiarització amb GlusterFS	16	65	1040 €
Testeig	16	65	1040 €
Alta Disponibilitat	16	85	1360 €
Implementació en EFS	16	75	1200 €
Documentació i presentació	16	50	800 €
		Total	6640 €

Taula 5.3 Costos humans

5.1.4 Costos indirectes

Tenim en compte costos indirectes com la llum, la connexió a internet o el lloguer de les oficines principalment, que són els que fan viables poder treballar amb el portàtil, connectar-se als servidors remots i recercar informació. El càlcul de kWh correspon al nombre d'hores del projecte junt amb el consum màxim del carregador per mantenir l'ordinador portàtil, 130W. La taula 5.4 conté les dades.

Producte	Preu	Unitats	Estimació
Llum	0.14€/kWh	54	7.56 €
Quota mensual d'internet	56 €	5	280 €
Quota mensual oficina	500 €	5	2.500 €
		Total	2.787.56 €

Taula 5.4 Costos indirectes

5.1.5 Costos totals

Si comptabilitzem tots els costos, tenim una aproximació del projecte corresponent a la següent taula, contant una contingència del 5% en ser un projecte ben controlat quant a costos, dins de la taula 5.5 ho podem visualitzar:

Producte	Estimació
Costos Hardware	161.10 €
Costos Software	0.38 €
Costos Humans	6640 €
Costos Indirectes	2.787.56 €
Contingència (5%)	479.45 €
Total	10.068.49 €

Taula 5.5 Costos totals

5.2 Control de gestió

Al final de cada tasca es comprovarà si s'ha endarrerit respecte a la planificació, encara que això no representa un cost directe per a l'estudiant. Dintre de totes les possibilitats, aquest projecte no comporta grans riscos de gestió de pressupost, en incorporar en les eines software gratuït en la gran majoria, i pocs recursos hardware. Comptant que el proveïdor de servidors ens ofereix un percentatge de funcionament per sobre del 99.9% i el portàtil té menys de 6 mesos, sent amb una marca fiable com a tecnicosclic.com [12] expliquen, assignem una contingència del 5%.

Per calcular les irregularitats al projecte, principalment a les possibles hores de cada tasca, tenim una fórmula que ens servirà per comprovar-les:

- Desviació en el cost d'hores: (hores estimades – hores reals) * cost per hora.

5.3 Sostenibilitat del projecte

Dintre de la sostenibilitat valorarem tres camps, la sostenibilitat econòmica, l'ambiental i la social, així poder avaluar quant sostenible és el treball.

5.3.1 Dimensió econòmica

El cost estimat és bastant senzill en involucrar pocs elements per a aconseguir portar a bon terme el treball, per tant el pressupost és bastant útil per veure que el vertader repte d'aquest projecte correspon a les mateixes habilitats de l'estudiant/treballador, i no a sortejar possibles imprevistos.

Actualment el problema que es vol resoldre, abans d'aplicar cap solució, se soluciona tenint a cada servidor un disc intern per emmagatzemar la informació. Però aplicant la solució corresponent al treball, s'aconseguiria un estalvi, en la compra de disc durs i en el manteniment d'aquests al reduir el número d'aquests creixent de forma lineal l'estalvi segons l'empresa creixi.

Addicionalment, en ser un desenvolupament basat en software, no dependent directament del hardware, la vida útil és gran al no requerir renovacions cada x anys. Per tant, el projecte és viable i útil en la dimensió econòmica.

5.3.2 Dimensió ambiental

El fet de reduir elements hardware aplicant la solució del treball implica reduir l'empremta ecològica de l'empresa. Com menys dispositius, menys elements necessaris per reciclar. Indirectament es redueix el consum elèctric al compartir el sistema de fitxers entre els servidors, disminuint així més encara l'impacte ecològic.

Respecte al desenvolupament del treball, el cost de recursos no és especialment gran. A més al poder treballar en remot el número de transports necessaris es redueix, disminuint la contaminació generada pel possible mitja de transport utilitzat.

5.3.3 Dimensió social

Aquest treball no té gran presència en aquest apartat, perquè si una empresa aplica aquest canvi, ho fa de manera transparent als usuaris finals. L'únic col·lectiu afectat per aquest treball és el personal encarregat de mantenir tot el hardware relacionat, al disposar de menys hardware que gestionar i facilitar les seves tasques. I pel desenvolupador del treball durant la producció del projecte. Els valors socials seran baixos a la matriu de sostenibilitat, amb l'únic punt on influeix lleugerament es als mencionats anteriorment.

5.3.4 Matriu de sostenibilitat

Basant-nos en els punts anteriors, podem valorar sobre 10 els següents punts, emplenant així la matriu de sostenibilitat del treball dins de la taula 5.6.

	PPP	Vida útil	Riscos
Econòmic	8	10	4
Ambiental	9	9	2
Social	3	2	0

Taula 5.6 Matriu de sostenibilitat

5.4 Legalitat del projecte

Com a projecte de l'àmbit de la tecnologia i la informàtica, i amb la recent entrada en vigor de la nova llei europea (Reglament General de Protecció de Dades, GDPR per les seves sigles en angles), és important que el projecte no vulneri cap apartat d'aquesta o les ja en vigor a espanya (Llei orgànica de protecció de dades, LOPD en sigles) més encara sent un possible component d'un producte per a una empresa.

En aquest projecte no ens hem de preocupar perquè totes aquestes lleis existeixen per a la protecció de les dades de l'usuari, unes dades que nosaltres no en tindrem mai doncs només emmagatzemarem programes per a la seva execució. Així doncs en cap cas incomplirem la LOPD o la GDPR.

6. Integració del coneixement

La integració del coneixement adquirit a diverses assignatures ha permès i facilitat el bon desenvolupament d'aquest projecte, principalment aquestes assignatures han sigut: Xarxes de computadors (XC/XC2), Projecte d'enginyeria de computadors (PEC), Sistemes de temps real (STR) i Centres de processament de dades (CPD).

De Xarxes de Computadors s'ha aplicat el coneixement d'organització i creació de xarxes, perquè la plataforma desenvolupada s'executa en xarxa. Alhora per aplicar solucions d'alta disponibilitat s'ha muntat un clúster en xarxa, configurant cada node dins de la xarxa, molt semblant als laboratoris d'aquesta assignatura.

Del Projecte d'Enginyeria de Computadors sobretot s'ha aplicat el coneixement sobre com desenvolupar un projecte i com organitzar-ho. El control de les dates límit, de forma que el projecte avenci de forma contínua i la presa de decisions en cas de possibles (o necessaris canvis) són una part important de qualsevol projecte. Els costos, i les amortitzacions per exemple són altres parts que es tracten a PEC que han ajudat al projecte.

De Sistemes de Temps Real principalment s'ha aplicat el coneixement al desenvolupament de l'alta disponibilitat al projecte. Aquesta part requeria especialment aplicar solucions per garantir que la solució del projecte sempre funciones. En STR s'aprèn a valorar el temps necessari perquè un sistema respongui a temps, i a mesurar i valorar si aquest és adient, doncs la solució del nostre projecte requereix un bon temps de resposta (però no arribant tant a l'extrem com un sistema de temps real).

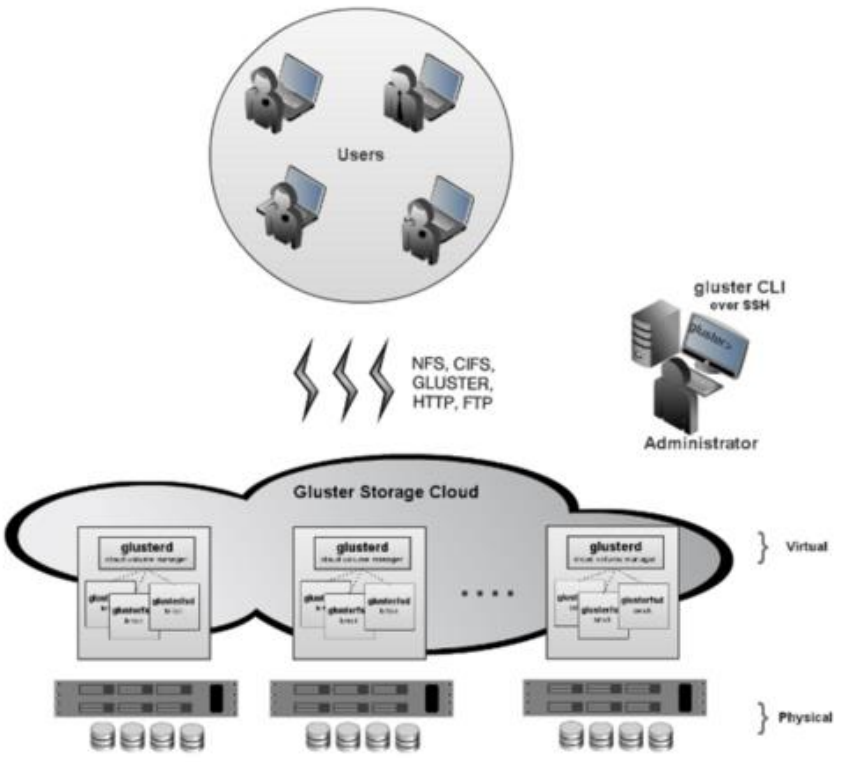
A Centres de Processament de Dades s'aprèn a valorar, investigar i decidir les característiques que ha de tenir un sistema per a complir els requisits. Aquesta part és molt important en el projecte, per valorar quins requeriments necessita el nostre projecte per tal de no utilitzar servidors massa potents, ni massa petits, alhora de l'amplada de xarxa adient. D'altra forma tindríem un sistema molt car per al rendiment que necessitem, o un massa barat però que no compleix els nostres requisits.

7. GlusterFS

Per solucionar el punt de la distribució de dades existeixen diverses solucions que serveixen per crear i distribuir un sistema de fitxers (podem veure exemples al punt 1.3). Nosaltres utilitzarem GlusterFS per ser el software que més s'adequa a les nostres necessitats.

Com ja hem vist anteriorment, GlusterFS es un sistema de fitxers distribuït de codi obert. Té capacitat per suportar fins a petabytes de dades, agrupant els seus anomenats *bricks,* dins dels servidors que conformen el clúster (o conjunt d'ordinadors units per treballar junts). GlusterFS crea un sistema de fitxers basat en directoris.

Com podem observar a la il·lustració 7.1, un escenari de GlusterFS estaria compost pel "núvol" creat per GlusterFS. Els clients que vulguin connectar-se al sistema de fitxers es connectarien a aquest núvol, disposen de vàries opcions per connectar-se com per exemple peticions HTTP, eines NFS, clients FTP (sigles de File Transfer Protocol, o Protocol de Transferència de Fitxers). La infraestructura física és transparent al client, i l'administrador del sistema configura l'accés al núvol com desitgi.



Il·lustració 7.1 Arquitectura de GlusterFS [2]

[2] https://docs.gluster.org/en/v3/Quick-Start-Guide/Architecture/ [17]

Aquest clúster pot contenir d'un a N servidors, al ser un sistema muntat per software no depèn del hardware. És a dir, no té cap limitació de components, pot funcionar allò on un sistema operatiu compatible estigui instal·lat. Ara bé, òbviament el rendiment màxim sí que pot variar segons les característiques de la xarxa, la càrrega de peticions, o la memòria dels servidors entre altres coses.

Per desplegar la primera versió de GlusterFS primer hem de decidir quin hardware necessitarem, perquè el proveïdor de servidors utilitzats per la nostra empresa ens ofereix solucions, de les quals hem d'escollir l'adequada.

7.1 Selecció del hardware

Segons Red Hat [18] (propietària de GlusterFS) els requeriments canvien segons l'ús que volem donar al sistema. A la taula 7.1 podem veure els requisits bàsics segons el tipus d'ús que li donem a GlusterFS, es divideix en tres casos:

	Computació d'alt rendiment	Ús general	Arxivar dades
Processador	Intel Xeon Nehalem-Ex	Intel Xeon Nehalem-Ex	Intel Xeon Nehalem-Ex
Memòria RAM	48GB	32GB	16GB
Disc	15000 RPM SAS (HDD) SSD	10.000 RPM (HDD) 7.200 RPM (HDD) SSD	10.000 RPM (HDD) 7.200 RPM (HDD)
Xarxa	2x10 Gbit	2x10 Gbit 2x1 Gbit	2x10 Gbit 2x1 Gbit

Taula 7.1 Requeriments GlusterFS

El nostre cas estaria en un punt més proper a l'ús general, perquè tampoc hem d'emmagatzemar fitxers de Terabytes com podria ser en el cas de la computació d'alt rendiment. Així doncs, dintre dels servidors del nostre proveïdor, un que s'ajusta ve al nostre sistema, i alhora té una bona proporció qualitat preu pot ser un amb:

32 Gigabytes
4 Cores Xeon
10 Gbit connexió
SSDs de capacitat variable.

7.2 Desplegament

7.2.1 Infraestructura desplegada

Per a tenir els fitxers distribuïts en diversos servidors per lògica necessitem més d'un servidor. Per al nostre projecte utilitzarem tres servidors (el sistema de fitxers estarà allotjat en aquests tres diferents).

El fet de mantenir les dades en més d'un servidor ens atorga la seguretat que en cas que un d'ells deixi de funcionar, les nostres dades estaran disponibles en l'altre servidor (o altres). Però això comporta un requeriment important, tots els servidors han de tenir sempre la informació actualitzada, en qualsevol altre cas es pot perdre informació si servidor el principal falla.

Amb GlusterFS, quan estem connectats a un dels servidors i fem una escriptura, aquesta es propaga a tots els servidors que conformin el clúster (des de 2 fins a N). D'aquesta forma tots els servidors tenen sempre la informació actualitzada, però que passa quan un node del clúster es desconnecta i es torna a connectar (independentment del motiu)?

Quan el servidor que ha caigut es torna a connectar, GlusterFS envia totes les dades pendents que no té actualitzades, així es torna a posar al dia. Fins aquest punt tot funciona correctament, però pensem l'hipotètic cas en què tenim un clúster amb dos servidors, anomenats A i B. A i B emmagatzemen la variable var, que val 1. Per un problema en la xarxa A i B no es poden comunicar entre ells:

Estat inicial:
Servidor A: var = 1 Servidor B: var = 1

Succeeixen els següents esdeveniments:
Servidor A: Modifica var, var = 2 (No es notifica a B el canvi per l'error de xarxa)
Servidor B: Modifica var, var = 3 (No es notifica a A el canvi per l'error de xarxa)

Estat final:
Servidor A: var = 2 Servidor B: var = 3

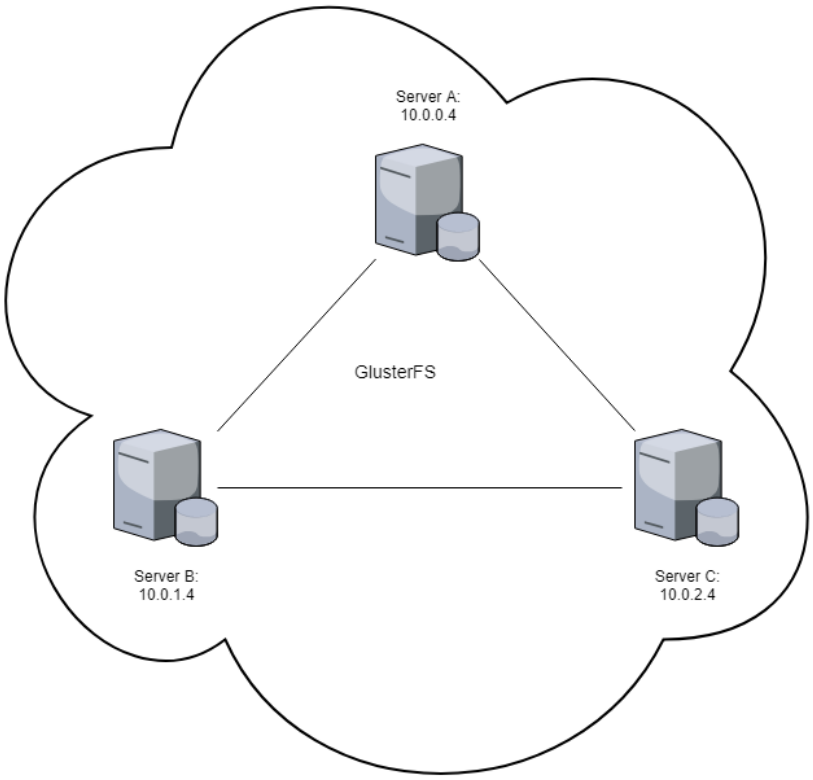
A causa del funcionament de GlusterFS, el qual no utilitza un node central per comunicar las escriptures, en perdre la connexió entre ells, cada servidor creu que l'altre ha caigut i segueix funcionant de forma independent. En tornar-se a connectar A creu que té l'última versió, i B també, llavors el sistema es queda bloquejat.

La situació ocorreguda a l'exemple s'anomena Split Brain. Afegint un tercer node ens assegurem que això no passi (però aquest tercer node ha de tenir un camí de xarxa diferent perquè sigui més segur, si té el mateix podem arriscar-nos a què un problema en aquesta xarxa també l'afecti).

Una alternativa a utilitzar un tercer servidor de dades seria utilitzar un servidor extern que actues com a servidor de votacions (quorum), així en cas que succeeixi una situació com l'anterior, abans de fer el canvi el servidor de votacions decidiria qui té preferència. De totes maneres, per al nostre cas hem escollit el tercer servidor de dades perquè ens és més útil tindre més seguretat quant a la nostra informació.

Per tant, utilitzarem tres servidors, connectats entre ells en una mateixa vpc (per evitar connexions alienes exteriors). GlusterFS està disponible per vàries distribucions de Linux, en el nostre cas utilitzarem Ubuntu 16.04 LTS, així tenim uns servidors amb un sistema operatiu amb suport garantit durant anys (fins a 2021 assegurat).

Cada servidor té un disc SSD secundari de 100GiB, on s'allotjarà el sistema de fitxers de GlusterFS. A la il·lustració 7.2 podem veure el mapa conceptual de la nostra xarxa, amb els 3 servidors, i les adreces IP utilitzades per al desplegament.



Il·lustració 7.2 Mapa de xarxa del projecte

7.2.2 Instal·lació de GlusterFS

Per instal·lar GlusterFS, la primera part del treball, s'ha seguit la documentació disponible (https://docs.gluster.org/en/latest/), amb únic canvi, aquesta documentació proporciona l'enllaç a una versió més antiga de GlusterFS, la 3.8.

Aquesta versió té un problema important, quan estem connectats a un node del clúster (per utilitzar el sistema de fitxers remot), i aquest node es reinicia (independentment del motiu), mai es pot tornar a connectar el client. Gluster mostra l'error "Transport adduint is not Connected". Utilitzant l'última versió de GlusterFS, la 4.0, l'error se soluciona. Considerem important comentar això no es menciona a la documentació i hi ha poca informació respecte al problema.

Els primers passos a seguir (a tots tres servidors del clúster) és instal·lar el software de GlusterFS, es pot fer amb les comandes següents:

```
> sou apt install -y software-properties-common
```

Aquesta comanda ens afegeix les comandes necessàries per afegir repositoris de software externs (no inclosos amb el sistema operatiu per defecte). En el cas de GlusterFS, és necessari.

```
> sudo add-apt-repository ppa:gluster/glusterfs-4.0
```

Important aquest pas, aquí afegim el repositori que conté GlusterFS per instal·lar-ho i és el pas que difereix de la documentació. Si volem seguir-la, la comanda acabarà en glusterfs-3.8, que és la versió amb el problema mencionat anteriorment. Únicament utilitzant glusterfs-4.0 afegirem el repositori amb l'última versió.

```
> sudo apt-get Update
```

Actualitzem el sistema una vegada afegit el nou repositori.

```
> sudo apt-get install glusterfs-server
```

I finalment instal·lem GlusterFS. El següent pas seria habilitar la comunicació de GlusterFS entre els servidors. Ho faríem de la següent manera:

```
> iptables -I INPUT -p all -s '<ip-address>' -j ACCEPT
```

On ip address correspondria a l'adreça IP dels servidors, aquest pas seria diferent a cada node, utilitzant el servidor A com a exemple:

```
> iptables -I INPUT -p all -s 10.0.1.4 -j ACCEPT
```
(Habilitar comunicació amb servidor B)

```
> iptables -I INPUT -p all -s 10.0.2.4 -j ACCEPT
```
(Habilitar comunicació amb servidor C)

No cal fer una amb la pròpia del servidor, perquè és redundant habilitar la comunicació amb un mateix en aquest cas.

Ara bé, aquest pas en el nostre cas no és l'adequat, perquè nosaltres hem de gestionar les opcions de xarxa directament amb el nostre proveïdor de servidors. Podem permetre qualsevol comunicació entre ells directament, o si volem ajustar més la seguretat permetre únicament la comunicació pels ports necessaris de GlusterFS.

Els ports que utilitza GlusterFS per comunicar-se són:

24007 TCP – Servidor local de GlusterFS

>49152 TCP – (Un per cada node del clúster) en el nostre cas del 49152 al 49154

2049 TCP – Per a la comunicació amb NFS als clients del sistema de fitxers

111 TCP/UDP – Per a la comunicació amb NFS als clients del sistema de fitxers

7.2.3 Configuració DNS (opcional)

En aquest pas, configurarem els servidors de manera que entre ells es puguin trobar directament amb el nom. Per exemple, en comptes d'utilitzar l'adreça IP 10.0.0.4, podrem utilitzar el nom Gluster1. Molt més còmode i visual, ja que ho utilitzarem moltes vegades. No és obligatori, perquè ens podem referir sempre amb les adreces IP, però és una bona pràctica, i recomanable.

En el cas d'Ubuntu hem de canviar el nom de cada servidor dintre del fitxer /etc/hostname, i afegir l'adreça dels altres al fitxer /etc/hosts. Podem escollir el nom que vulguem, en aquesta xarxa farem servir gluster1, gluster2, i gluster3.

El fitxer *hostname* conte el nom del servidor propietari del fitxer, per tant el servidor A conte "gluster1", el servidor B conté "gluster2" i finalment el servidor C conté "gluster3".

En canvi, el fitxer *hosts* conté les traduccions locals d'adreces (en diem locals per que si no apareixen aquí, es tradueixen igual, però s'encarrega un servei extern). els nostres servidors puguin contactar amb els noms dins de la xarxa local, hem d'escriure a quina adreça IP correspon cada nom.

Per defecte el fitxer hosts només conté la traducció *local host*, que es tradueix com l'adreça local per tornar al host del fitxer com diu el nom. Nosaltres afegim les noves línies, quedant-nos el fitxer així en tots tres servidors:

```
127.0.0.1  localhost #Aquesta adreça es la comentada anteriorment
10.0.0.4   gluster1
10.0.1.4   gluster2
10.0.2.4   gluster3

#configuració per defecte d'IPv6, aliena al projecte
::1        localhost ip6-localhost ip6-loopback
ff02::1    ip6-allnodes
ff02::2    ip6-allrouters
```

Una vegada fet per als tres servidors podem o reiniciar el servei encarregat de gestionar el hostname, o simplement reiniciar-los. Podem comprovar si funciona correctament fent *ping* (envia un missatge, si respon el destinatari, hi ha comunicació) des d'alguna màquina a una altra amb el nom, i no amb la IP, per exemple des de gluster1:

```
> ping gluster2
> ping gluster3
```

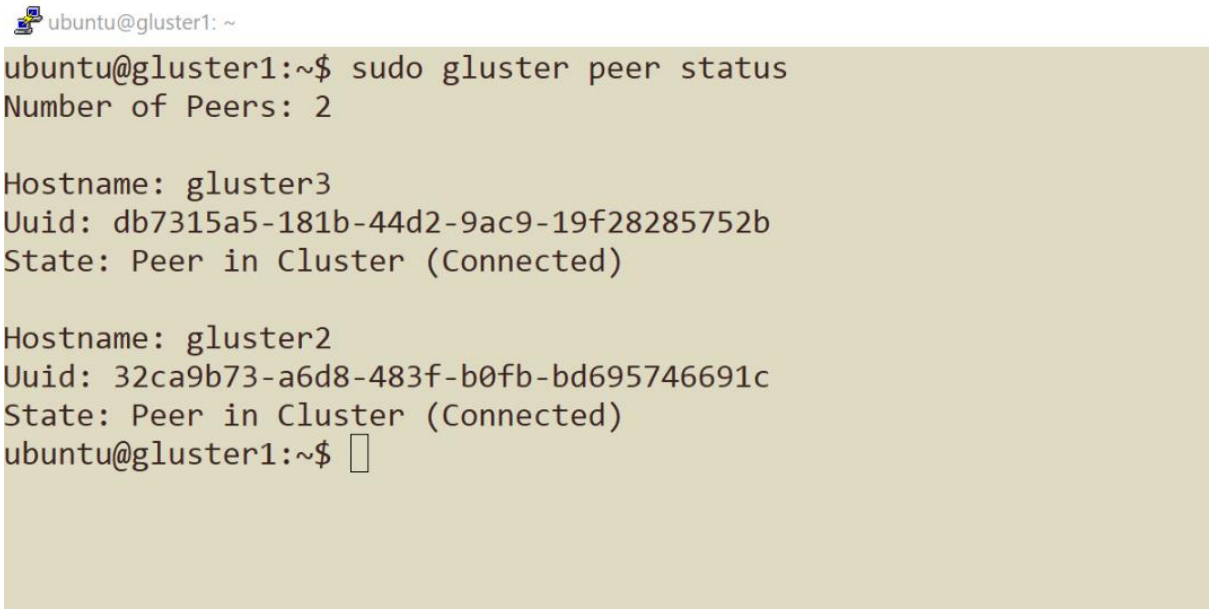
Cal dir que, si tenim una xarxa restrictiva, hem de permetre el tràfic ICMP, que és el protocol amb el qual funciona l'eina *ping.*

7.2.4 Creació del clúster

Ara que tots els nodes es poden comunicar entre si, hem de crear el clúster que conformarà el sistema de fitxers. Amb GlusterFS es fa amb l'eina probe, es pot fer des d'un sol servidor, doncs en iniciar la comunicació amb la resta aquests rebran també la configuració del clúster. Nosaltres ho farem des del servidor A, per tant hem de fer:

```
> gluster peer probe gluster2
> gluster peer probe gluster3
```

Com veiem, no cal fer probe a un mateix. Si volem comprovar que el clúster s'ha creat correctament podem utilitzar l'eina status de Gluster. L'eina status ens retorna l'estat del clúster des del punt de vista del servidor on s'executa la comanda, a la il·lustració 7.3 podem veure un exemple donant-nos el nombre de servidors connectats (peers), el nom, l'identificador i l'estat.



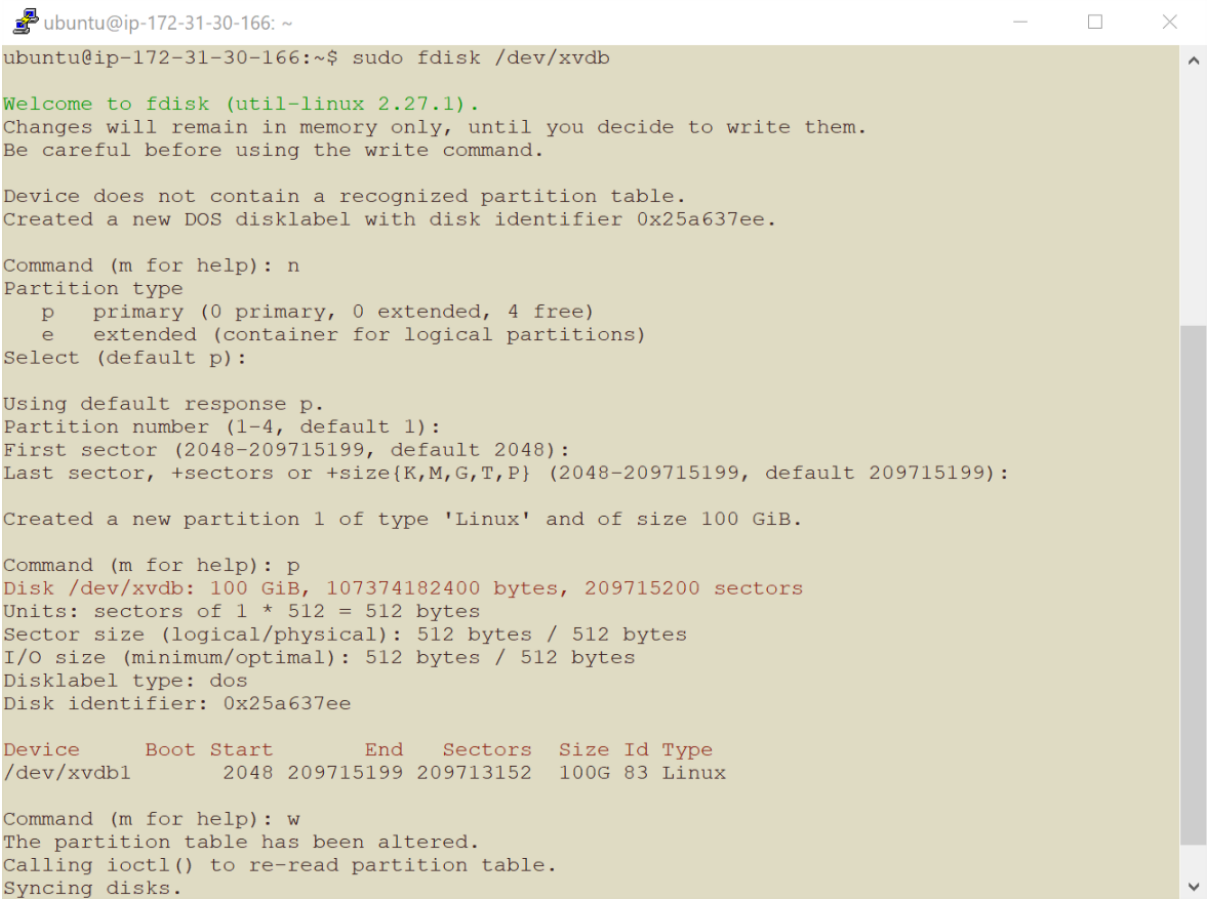
*Il·lustració 7.3 Resultat de la comanda des de **gluster1***

7.2.5 Creació del disc remot

Una vegada el clúster ja està connectat, podem crear el disc. En el nostre cas, els servidors tenen un disc secundari de 100GiB (ja mencionat abans), però ve sense cap format ni particions. Per tant abans de poder utilitzar-ho ho hem de donar-li format.

Primer utilitzem l'eina *fdisk* per crear una partició al disc, en el nostre cas localitzat a /dev/xvdb. A causa de ser diferents passos ho mostrem a la il·lustració 7.4, on veurem els següents passos:

1. Cridem a l'eina fdisk amb: `sudo fdisk /dev/xvdb`.
2. Creem una nova partició amb l'opció '`n`', amb els consegüents valors per defecte.
3. Visualitzem la nova partició amb l'opció '`p`'.
4. Si tot es correcte, guardem els canvis amb l'opció '`w`'.



Il·lustració 7.4 Preparació del disc amb fdisk

Ara que el nostre disc ja té una partició cal donar-li format. Seguint la recomanació de la documentació de GlusterFS, li donarem al disc format XFS. XFS. Això és degut al fet que GlusterFS està molt més testejat sota XFS, a més hi ha més errors coneguts utilitzant l'altre format més comú, ext4.

A més, es recomana que la mida de l'inode (l'estructura de dades que emmagatzema la informació relativa als fitxers i directoris) sigui mínim de 512 bytes. Això és degut al fet que per defecte XFS utilitza llistes de control d'accés (ACL's en anglès), i si aquestes llistes ocupen més de la mida de l'inode, cada accés requerirà més d'una consulta, sent menys eficient.

Com realitzem aquest format amb aquestes característiques? Amb la següent comanda:

```
> mkfs.xfs -i size=512 /dev/xvdb1
```

L'opció -i correspon a les opcions de l'inode, que en el nostre cas és per modificar la mida, seguit de la ubicació de la partició, /dev/xvdb1.

Ara afegim la nova partició amb format al fitxer /etc/fstab [13], aquest fitxer s'utilitza per definir com es munten les particions (entre altres coses) al sistema. Una entrada nova a /etc/fstab es compon de les següents parts:

File system: Defineix la partició o dispositiu d'emmagatzemament a muntar.
Dir: Correspon a la ruta on serà muntada la partició.
Type: Indica el tipus de sistema de fitxers, en el nostre cas xfs.
Options: Permet indicar opcions de muntatge en cas necessari.
Dump: Utilitzat pel programa dump, si és 1, es crearà una còpia de seguretat.
Pas: Indica l'ordre de comprovació, 0, 1 o 2, per al programa fsck:
0: Sense comprovació. 1: Alta prioritat, només pel sistema arrel.
2: La resta de sistemes que es vulguin comprovar.

Seguint l'ordre en què les hem descrit, la nostra entrada a /etc/fstab seria:

```
<file system>       <dir>              <type>       <options> <dump> <pass>
/dev/xvdb1          /data/gluster      xfs          defaults  0      0
```

Per tant, per afegir la línia al fitxer, podem utilitzar la comanda echo:

```
> echo "/dev/xvdb1 /data/gluster xfs defaults 0 0" >> /etc/fstab
```

El directori el qual hem escollit per muntar és /data/gluster, com podem veure a la línia que hem afegit a /etc/fstab. Per tant, hem de crear aquest directori.

```
> mkdir -p /data/gluster
```

L'opció -p crea tots els directoris necessaris anteriors, per si no existeixen, en aquest cas /data per exemple. Ara que tenim la carpeta llesta, muntem la partició, podríem reiniciar el servidor i que s'apliquin les opcions del /etc/fstab, o executar la comanda:

```
> mount -a
```

Que essencialment executa la configuració que conté el fitxer /etc/fstab, sense necessitat de reiniciar. Ara creem una carpeta dintre per a GlusterFS, ja que no pot estar directament a l'arrel d'on s'ha muntat la partició.

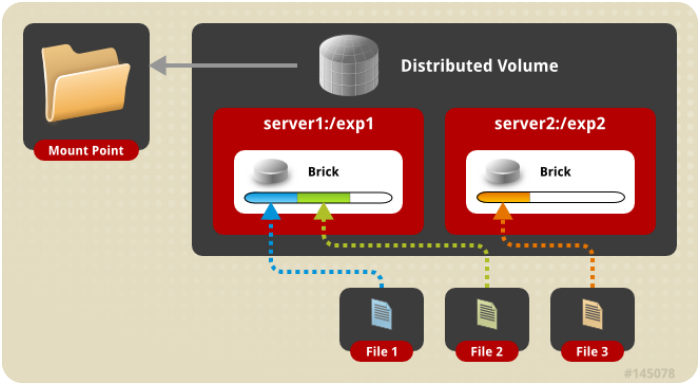
```
> mkdir -p /data/gluster/brick
```

Ara que tenim el directori i el disc llest, creem el volum de GlusterFS, la següent comanda, com totes les de GlusterFS, s'executa en un dels servidors, no en els tres (és tota una línia, no varies separades):

```
> gluster volume create brick replica 3
      gluster1:/data/gluster/brick
      gluster2:/data/gluster/brick
      gluster3:/data/gluster/brick
```

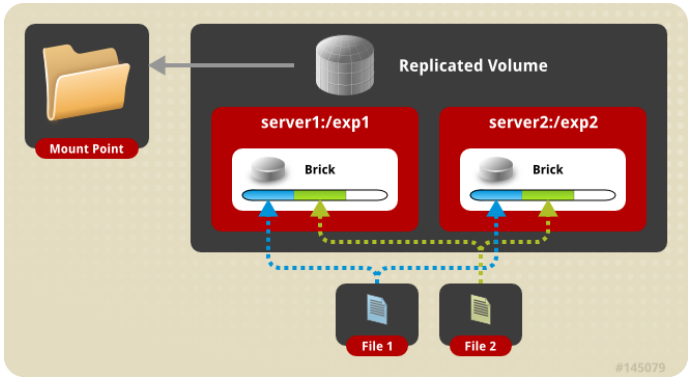
Aquesta línia indica vàries coses, la comanda principal és *"gluster volume create"*, tot seguit ve el nom que volem donar-li al volum, en el nostre cas *"brick"*. El següent paràmetre *"replica"* indica el tipus de volum, a GlusterFS existeixen (no obsolets):

- Distribuït: Les dades es reparteixen per tots els servidors del clúster, en el nostre cas amb tres servidors, si creéssim tres fitxers, cadascú d'ells aniria a un servidor diferent. Podem veure un exemple del volum a la il·lustració 7.5.



Il·lustració 7.5 Exemple de volum dispers[3]

- Replicat: La informació es replica a tots els servidors, ofereix com a avantatge el fet de multiplicar la màxima velocitat de connexió a l'agrupar tots els servidors, a més de seguretat en cas de caiguda d'un servidor (en un volum distribuït perdríem la informació distribuïda en aquest servidor caigut). Podem veure un exemple del volum a la il·lustració 7.6.



Il·lustració 7.6 Exemple de volum replicat [3]

[3] https://docs.gluster.org/en/v3/Quick-Start-Guide/Architecture

- Dispers: Cada fitxer se separa en diverses parts juntament amb informació de recuperació. Això permet que en cas de caiguda d'un servidor, es pugui recuperar la informació amb els codis que contenen els altres servidors. El nombre de servidors necessaris per recuperar la informació es configura en el moment de crear el volum.

- Dispers distribuït: Combina el fet de distribuir fitxers amb la dispersió dels fitxers i la informació creada en el dispers per recuperar la informació en cas de perdre algun servidor.

Primerament es va provar el volum dispers, però durant el test del volum es va observar com diverses aplicacions es comportaven errròniament, o inclús no funcionaven. Aquest error no va ocórrer utilitzant un volum replicat, a més disposem de millor protecció contra caiguda i diferents servidors per accedir a la informació en cas necessari.

Finalment, l'últim paràmetre, *"3"*, indica el número de servidors que conformaran aquest volum, perquè encara que el clúster ja sigui creat, podem crear diferents volums i potser no ens interessa utilitzar tots els servidors d'aquest clúster. Els 3 servidors s'indiquen a continuació del número, seguint el patró <Servidor>:<Directori on s'allotjarà el volum>. El servidor pot indicar-se amb una adreça IP, o com nosaltres hem configurat el DNS prèviament, amb el nom.

Com a últim pas del volum, s'ha d'iniciar, executant:

```
> gluster volume start brick
```

On li diem que inicialitzi el volum anomenat *"brick"*. Si la comanda s'executa correctament, veurem un missatge de "`Starting brick has been successful`".

Si volem comprovar que tot funciona correctament podem executar les comandes *"heal"* o *"info"*.

`> gluster volume heal <nom del volum> info:` Ens mostrarà si algun volum necessita reparar-se (ja que no està sincronitzat per exemple). Aquesta operació mai ens farà falta al principi, perquè el volum estarà buit. En cas de necessitar efectuar l'operació de reparació, s'ha d'executar la comanda sense el paràmetre *"info"*.

`> Gluster volume info <nom del volum>:` Amb aquesta comanda sabrem la informació relativa al volum, el seu estat, i la seva configuració.

7.3 Configuració del volum

Els volums de GlusterFS es poden configurar amb diferents paràmetres. En el nostre cas, al ser bastant concret i fora del cas normal de GlusterFS, s'haurà de configurar el volum, ja que per defecte no podrem connectar-nos des d'un ordinador amb sistema operatiu Windows.

Des de Windows tenim dues opcions per connectar-nos:

- NFS: Només en versió 3 (Windows només suporta nativament actuar com a servidor versió 4, i no com a client). Encara que no és l'opció que recomanen a la documentació de GlusterFS, serà l'opció que utilitzarem pel fet que ofereix un rendiment bastant superior respecte a l'altre opció.

- CIFS: Molt més conegut com a Samba, és l'opció que recomanen a GlusterFS per muntar a Windows, però el rendiment és més dolent que utilitzant NFS [19], per tant, a causa dels nostres requeriments, no ens serveix.

Per defecte, GlusterFS no utilitza NFS. Llavors el que hem de fer és fer que el volum utilitzi NFS. Per això hem d'executar les següents comandes.

```
> gluster volume set <nom del volum> nfs.disable off
```

Amb aquesta comanda activem NFS, per defecte ve desactivat (encara que sigui una mica ambigua la nomenclatura, ve com "disable on").

```
> gluster volume set <nom del volum> nfs.volume-access
  read-write
```

Configurem els permisos d'accés per NFS com a lectura escriptura, per defecte pot vindre com "només lectura", així ens assegurem.

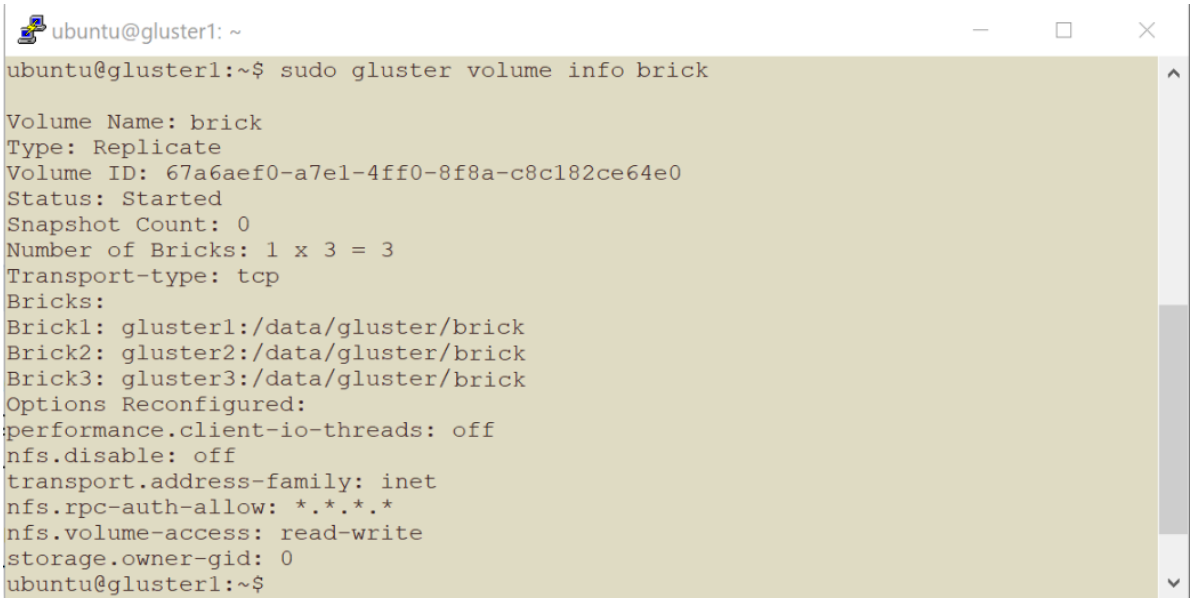
```
> gluster volume set <nom del volum> nfs.rpc-auth-allow <IP>
```

Amb aquesta comanda permetem l'accés al volum des de l'adreça IP que vulguem, en el nostre cas hauríem d'afegir els ordinadors que volem que es connectin al volum.

```
> gluster volume set <nom del volum> storage.owner-gid 0
```

Finalment assignem el propietari del volum a 0, l'usuari *root* d'Ubuntu, ho necessitarem configurar també en el registre de Windows.

Ara, si executem la comanda *"info"* mencionada anteriorment, podem veure com la configuració està ben posada. A la il·lustració 7.7 podem veure un exemple.



*Il·lustració 7.7 Exemple de **info** amb la configuració*

Com a últim pas **important**, cal remarcar que durant el desenvolupament del treball, es va trobar que Ubuntu 16.04 té un *bug* que **impedeix que el servei rpcbind s'iniciï correctament**, aquest servei és l'encarregat de fer funcionar el servei NFS, i si no s'inicia, no ens podrem connectar. Es pot arrancar manualment a cada servidor, per quan es reiniciïn tornarà a no funcionar. Aquest error es pot solucionar executant la següent comanda:

```
> systemctl add-wants multi-user.target rpcbind.service
```

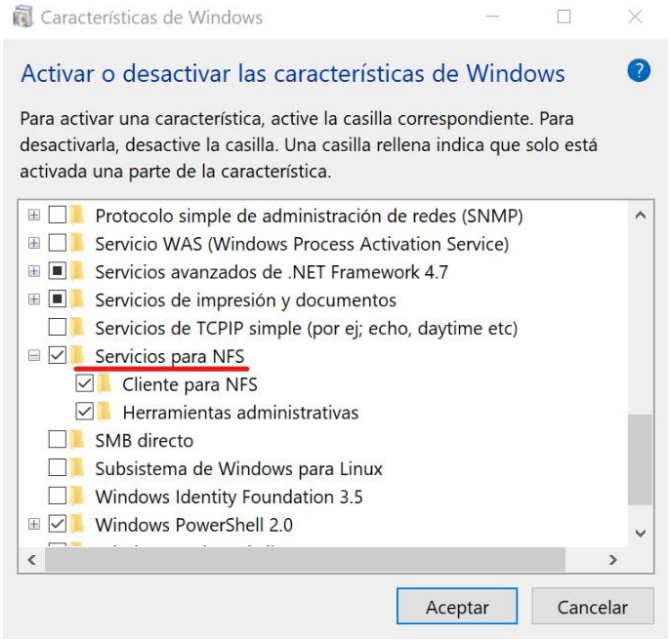
D'aquesta forma forcem que el servei arranqui per a tots els usuaris del servidor. Una vegada fet això, el servei rpcbind s'iniciarà bé (cal executar la comanda una vegada a cada servidor i no només en un, no és un component de GlusterFS).

7.4 Connexió Windows

Per connectar-nos des d'un client Windows, hem de modificar el registre de Windows, perquè utilitzi l'identificador que hem aplicat a la configuració del volum (la comanda que afectava el component **storage.owner-gid 0**).

Si el nostre sistema Windows no té els *"NFS Services"*, els hem d'instal·lar, doncs sinó el camp necessari al registre no existirà. Per instal·lar aquest component, hem d'entrar a l'apartat "programes i característiques" de Windows, i fer clic a "Activar o desactivar les característiques de Windows". Dintre del menú que ens apareixerà, hem d'activar

l'opció "Serveis per NFS" (per orientar-nos, a la il·lustració 7.8 podem veure exactament a quina característica ens referim). Aquesta opció no la tenen les versions *home* de Windows.

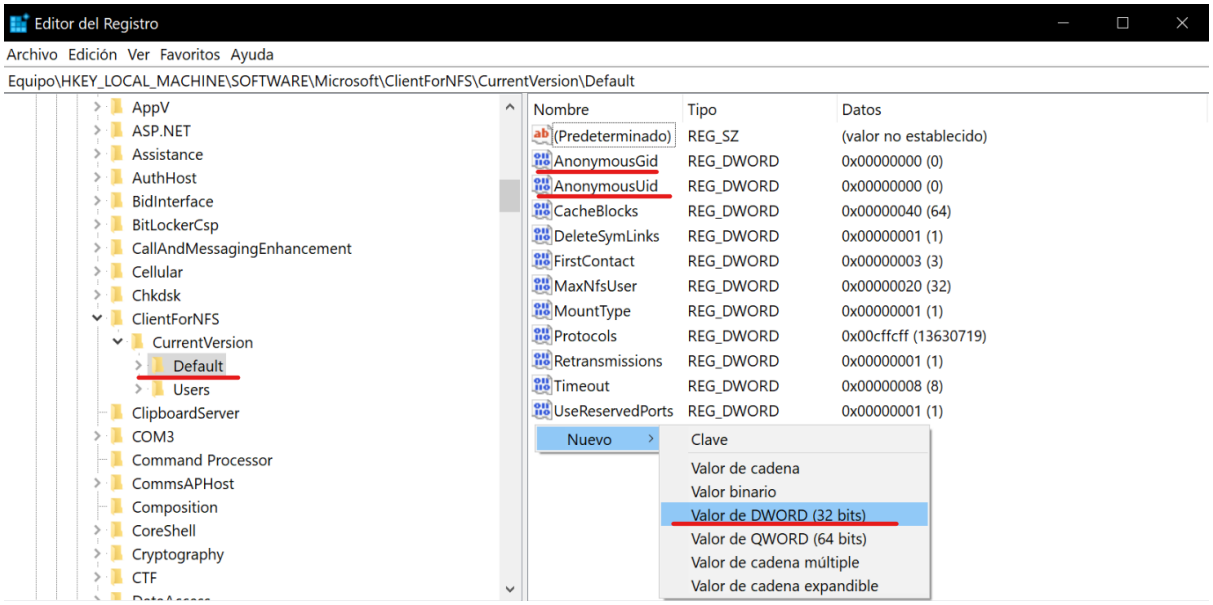


Il·lustració 7.8 Com activar els serveis per NFS

Una vegada activats cal modificar el registre de Windows. El registre de Windows és una col·lecció de dades que utilitza Windows per emmagatzemar configuracions de tot tipus, inclús les del servei NFS que necessitem modificar. Podem veure més informació al respecte a la següent adreça: https://support.microsoft.com/es-es/help/256986/windows-registry-information-for-advanced-users.

Podem obrir el registre de Windows buscant l'aplicació *regedit* al buscador de Windows. Ens haurem de dirigir a la adreça de registre:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\ClientForNFS\CurrentVersion\Default

Una vegada dins, haurem de crear dues entrades noves, del tipus DWORD (per crear-les només cal fer clic dret). La primera és "AnonymousUid", la segona "AnonymousGid". Ambdues han de contenir el valor que vam assignar al volum de GlusterFS, en el nostre cas 0. A la il·lustració 7.9 podem veure el registre de Windows, juntament amb les entrades comentades.



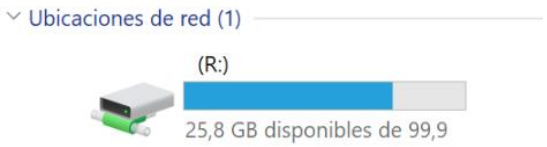
Il·lustració 7.9 Entrades del registre i com crear-les

Ara amb tot llest, l'últim pas és muntar el disc en el servidor Windows. Obrint la consola de Windows, podem executar:

```
> mount -o anon <Gluster Server IP>:/brick R:
```

Muntant amb l'opció *anon*, Windows agafa les opcions que hem configurat al registre de Windows. Hem de passar com a servidor qualsevol dels tres (en el nostre cas), i la lletra on volem muntar a Windows el disc, pot ser qualsevol que no sigui ocupada. Cal dir que encara que també es pot utilitzar la consola *powershell* (en aquells ordinadors Windows que la tinguin), la comanda *mount* executada no és la mateixa que a la consola original de Windows, per tant no funcionarà igual.

Una vegada executat, podrem utilitzar-ho com si fos un dispositiu més, en el nostre cas com ja hem fet proves amb ell, ja té espai ocupat. Com a detall, si naveguem amb l'explorador de Windows, podrem veure el disc fins i tot amb la informació d'espai com es veu a la il·lustració 7.10.



Il·lustració 7.10 Disc remot en l'explorador de Windows

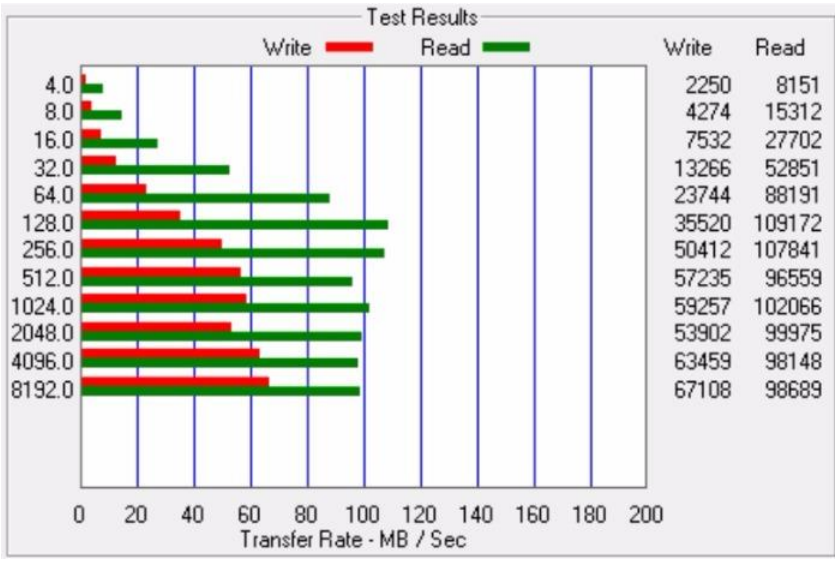
7.5 Tests i anàlisi de rendiment

Ara si, ja tenim el disc funcionant per tant podem realitzar les proves per validar el seu funcionament i analitzar el rendiment.

Per analitzar la velocitat efectiva utilitzem l'eina ATTO Benchmark [24], una eina que analitza la velocitat d'escriptura i lectura amb diferents mides de bloc (parts en les quals es divideix el fitxer a escriure o llegir, més mida de bloc, es dividirà en parts més grans).

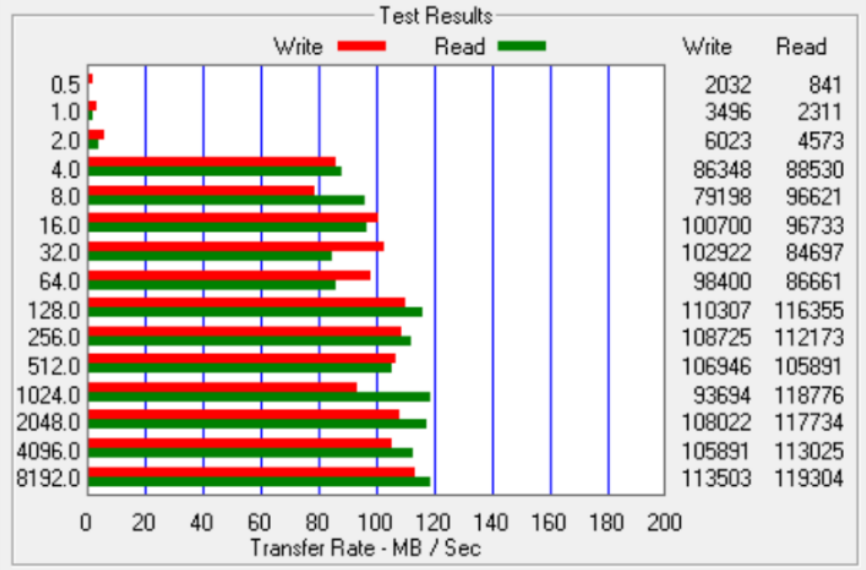
Per analitzar la latència executarem una aplicació pesada, en aquest cas el videojoc DOOM i mesurarem el temps necessari per ser llançat. El joc es considerarà llançat quan el primer frame es mostri. Com a unitat de referència, en els mateixos servidors de forma local, aquest temps ha sigut de 25 segons.

ATTO Benchmark ens retorna una gràfica. En aquesta gràfica l'eix vertical indica la mida de l'escriptura (vermell) o lectura (verd). L'eix horitzontal indica la velocitat en Megabytes per segon. Finalment a la columna dreta podem veure exactament la velocitat amb més resolució, Kilobytes per segon, per major claredat.



Il·lustració 7.11 Resultat amb GlusterFS

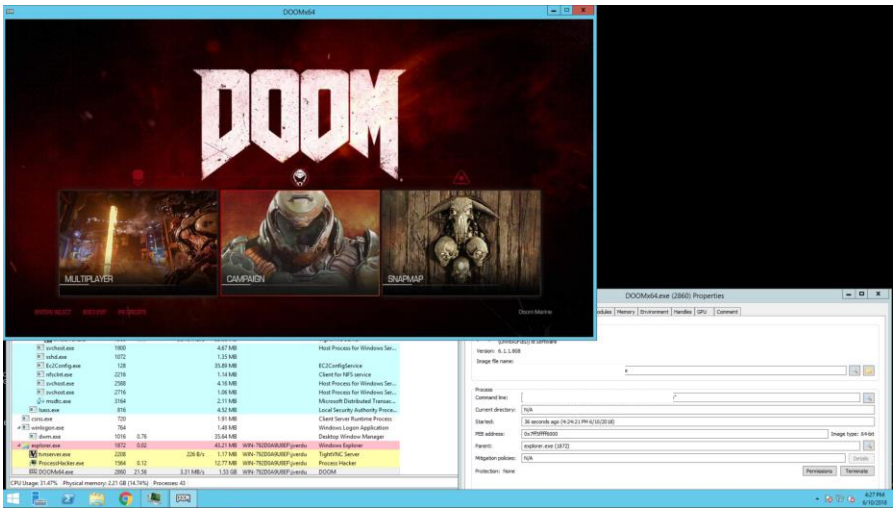
Com podem observar a la il·lustració 7,11, la lectura és bastant més ràpida que l'escriptura. Encara i així, els valors que observem són bastant bons (inclús en algun punt arriba a 110 MB/s). Per fer-nos una idea, podem veure el mateix test executat sobre un disc dur (no SSD) Western Digital Caviar Blue 1TB a la il·lustració 7.12:



Il·lustració 7.12 Resultat amb HDD

La diferència principal és a la velocitat d'escriptura, però el nostre sistema està pensat especialment per executar aplicacions (i llegir-les del sistema de fitxers remot), no tant per escriure.

Per fer una prova intensiva i mesurar la latència, executem el videojoc DOOM (ocupa més de 60GB).



Il·lustració 7.13 DOOM Executat en disc remot

Com podem veure a la il·lustració 7.13, s'ha executat sense cap mena de problema al sistema de fitxers remot, i s'ha llançat bastant de pressa. Exactament el primer frame s'ha mostrat al cap de 22 segons, 3 segons més ràpid que en local.

8. Alta disponibilitat

Abans de començar, hem d'entendre que és l'alta disponibilitat. Per entendre que és l'alta disponibilitat, primer definim la disponibilitat.

- Disponibilitat: Terme utilitzat per descriure el període de temps on un servei està disponible, a més del temps necessari perquè aquest sistema respongui a una petició. Es pot mesurar en percentatge, per exemple, un sistema amb disponibilitat del 50% funciona 1 de cada 2 dies, o 1 de cada dues hores (de mitja).

- Alta disponibilitat: Qualitat d'un sistema que assegura un alt nivell de temps funcionant, és a dir, una gran quantitat de disponibilitat.

Per tant, el que volem aconseguir és que el nostre sistema estigui disponible la major part de temps possible, i que si deixa de funcionar pel motiu que sigui, torni a funcionar l'abans possible. En el nostre cas, utilitzarem les eines Corosync [20] i Pacemaker[21]. Així doncs, que són?

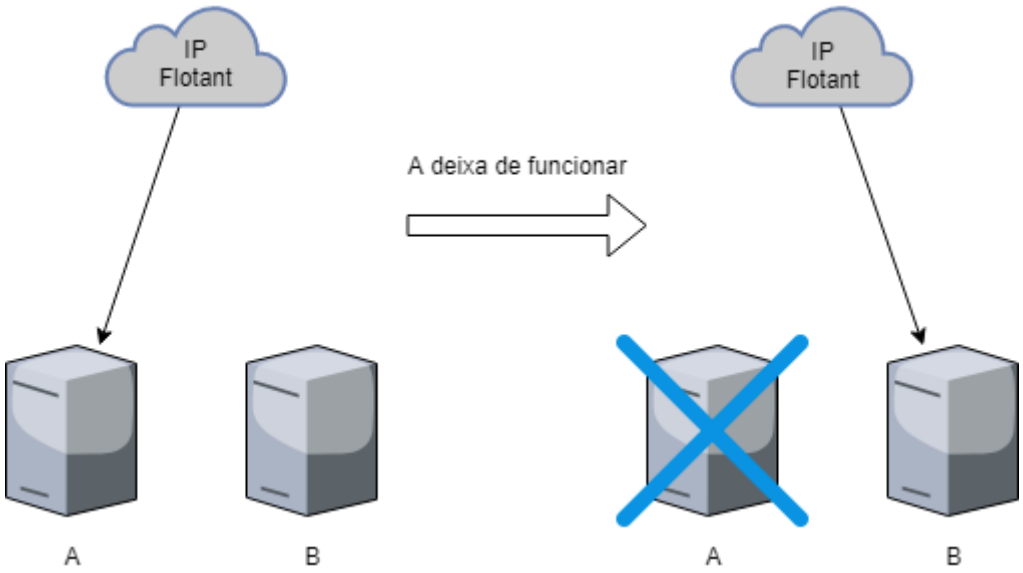
- Corosync: Eina open source, destinada a crear un sistema per gestionar clústers i permetre la comunicació entre els components del clúster. El servei corre de fons als servidors, monitorant el clúster.

- Pacemaker: Successor de HeartBeat, és una eina per gestionar recursos dintre d'un clúster, si Corosync s'encarrega de gestionar el clúster, Pacemaker comprova els serveis que tu programis dintre del clúster (per exemple si un servidor està caigut, o si NFS ha deixat de funcionar).

Per tant, Corosync s'encarregarà d'organitzar el clúster, i Pacemaker comprovarà que els servidors estiguin funcionant, i en cas de caiguda solucionarà els problemes perquè segueixi funcionant.

Com farem que en cas de caiguda, el servei segueixi funcionant? tipus de volum de GlusterFS, replicat, la informació és la mateixa a tots els servidors. Doncs, la solució seria que els clients es connectin a qualsevol dels altres servidors. Això s'ha de fer amb una IP flotant. Una IP flotant és una IP que canvia de servidor, però segueix sent la mateixa. Per veure-ho de forma més visual, a la il·lustració 8.1 podem veure un exemple.

Per exemple, si tenim una IP flotant 1.1.1.1, l'assignem al servidor gluster1. Si gluster1 cau, la IP s'assigna a gluster2 o gluster3. D'aquesta forma els clients només noten una desconnexió momentània, però després seguirà funcionant apuntant al nou servidor.



Il·lustració 8.1 Esquema d'IP flotant

8.1 Instal·lació de Corosync i Pacemaker

Per instal·lar els dos components, únicament hem d'executar la comanda següent (aquesta comanda s'ha d'executar en cada servidor que compongui el clúster):

```
> sudo apt install pacemaker pcs
```

El paquet pacemaker ja inclou el software Corosync, pcs és un software per configurar ambdues eines en conjunt, facilitant el procés.

Com són serveis que es comuniquen per la xarxa, si la nostra xarxa està protegida, hem de permetre el tràfic pels ports que utilitzin els serveis que volem, en aquest cas són:

- 2224 TCP: Permet la comunicació entre els pcs de tots els nodes.
- 3121 TCP: Utilitzat per Pacemaker quan hi ha un servidor fora de la xarxa local, un node remot.
- 5405 UDP: Utilitzat per Corosync en tots els nodes.

Ara bé, Corosync utilitza la comunicació multicast pel clúster, però la xarxa del nostre proveïdor de servidors, AWS, no permet multicast. Aquest fet, junt amb no poder gestionar directament cap IP flotant ens presenta el problema més difícil del projecte. Per poder solucionar aquest problema, hem de fer funcionar Pacemaker juntament amb la AWS. AWS disposa d'un software, la consola *cli*, que permet interactuar amb els

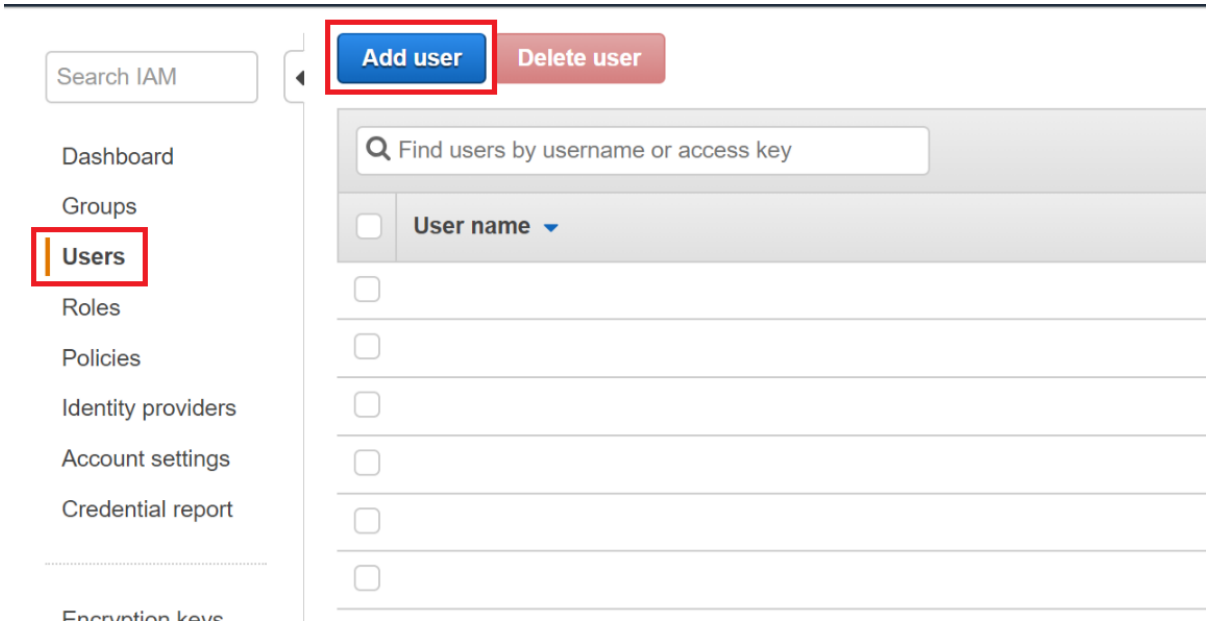
seus serveis des de servidors. D'aquesta manera podrem fer "flotar" una IP entre tots els servidors del clúster.

Per fer-ho, hem d'instal·lar la consola *cli* als servidors. Ho podem fer amb:

```
> apt install awscli
```

Ara, abans d'avançar, hem de preparar la configuració dins d'AWS. Per poder utilitzar la consola hem de donar-li un usuari d'AWS (perquè pugui interactuar amb els serveis del nostre compta). En el nostre cas creem un usuari especialment per això, d'aquesta manera només tindrà els permisos pel que ha de fer, sense arriscar res.

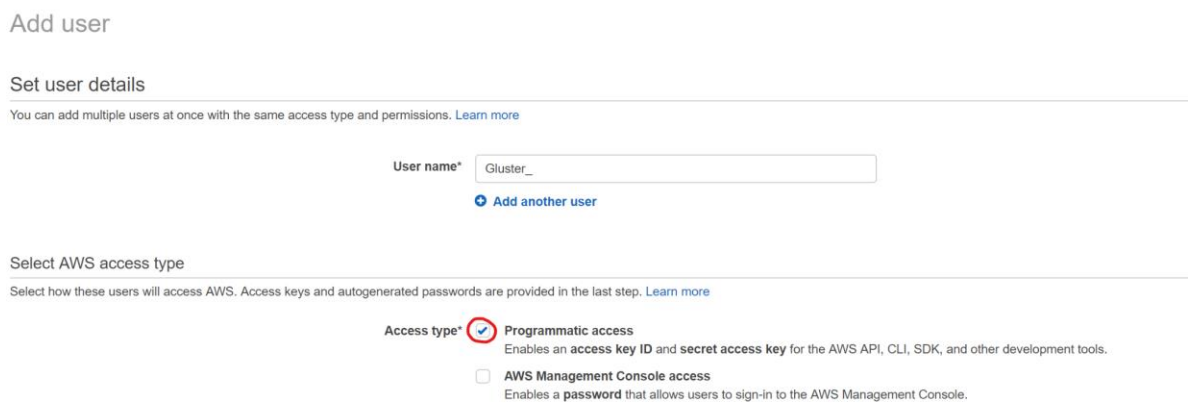
Per crear l'usuari, accedim al nostre compte d'AWS i entrem a l'apartat IAM, el servei d'AWS encarregat de gestionar els comptes d'usuari. Dintre de IAM, seleccionem l'opció de crear un nou usuari, dintre de l'apartat *Users* com podem veure a la il·lustració 8.2.



Il·lustració 8.2 Opcions del menú IAM

Ara, dintre de la creació de l'usuari, hem d'habilitar **l'accés programàtic com es mostra a la il·lustració 8.3**, és molt important, doncs només d'aquesta forma ens servirà aquest usuari per a la consola *cli*.

Durant la creació de l'usuari ens donaran una clau pública i una clau secreta (així les anomena AWS). És molt important sobretot que ens apuntem la Secreta, perquè sortir ja no la podrem tornar a saber, i haurem de reiniciar-la si la perdem.



Il·lustració 8.3 Accés programàtic

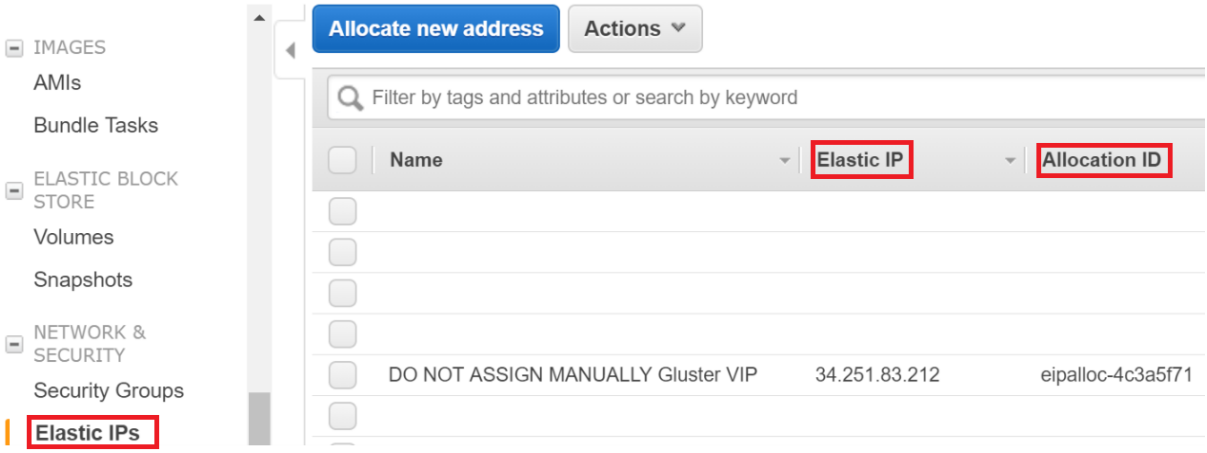
Una vegada ja tenim les dades de l'usuari per la consola, hem d'assignar-li els permisos necessaris perquè pugui manipular les adreces IP. Per fer-ho, hem de crear una política de seguretat. Ho podem fer des del mateix servei, IAM, anant a l'apartat *Policies*.

Dintre de *Policies* hem de crear una nova, seleccionant el botó blau *Create policy*. Aquest botó és a la mateixa posició i és del mateix tipus que el boto per crear un nou usuari. Una vegada dins, seleccionem l'opció JSON (per configurar la política manualment, encara que ho podríem fer amb l'editor visual). La política que necessita el nostre usuari per treballar amb les adreces IP la podem veure a l'annex A.1.

Aquesta política permet, veure quines direccions disponibles hi ha, descriure les regions que utilitzem a AWS, i els servidors que tenim. Juntament amb associar i des associar IP *Elàstiques* (nomenclatura utilitzada a AWS per a les adreces IP estàtiques), i fer el mateix amb adreces IP privades, sobre qualsevol servidor.

Una vegada creada la Política, tornem a l'apartat *Groups*, seleccionem el nostre usuari per a la consola (cada usuari té un grup per defecte, es crea automàticament), i fent clic a *Attach policy* podem afegir la nova política simplement buscant-la pel nom que li donem en el pas de crear-la.

Com a últim pas a AWS, necessitarem una *Elastic IP* per poder utilitzar-la com a flotant, aquesta la podem crear a l'apartat EC2 > Elastic IPs, seleccionant l'opció Allocate new address. De la nostra nova I necessitarem apuntar l'adreça IP, i l'Allocation ID. A la il·lustració 8.4 podem veure on són aquests camps per major claredat.

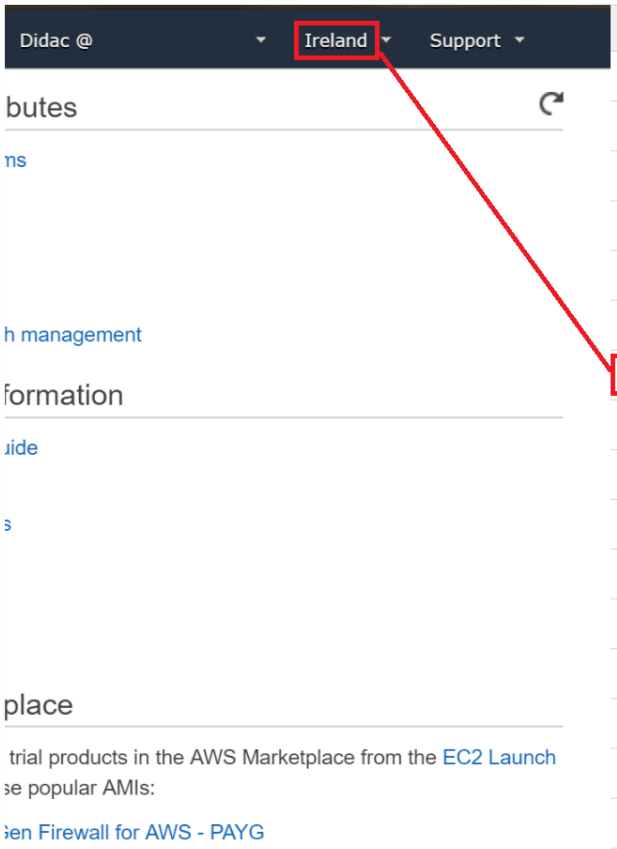


Il·lustració 8.4 Menú ElasticIP

Ara, amb l'usuari nou ens dirigim al servidor, i executem la comanda:

```
> aws configure
```

D'aquesta forma entrarem les dades necessàries per fer funcionar la consola *cli*. Podem deixar totes les dades per defecte amb excepció de la clau Pública, la Secreta i el codi de la regió on volem treballar amb el compte. A la il·lustració 8.5 podem consultar el codi de cada regió, a més de veure on es.



Code	Nombre
us-east-1	US East (N. Virginia)
us-east-2	EE.UU. Este (Ohio)
us-west-1	EE.UU. Oeste (Norte de California)
us-west-2	EE.UU. Oeste (Oregón)
ca-central-1	Canadá (Central)
eu-central-1	UE (Fráncfort)
eu-west-1	UE (Irlanda)
eu-west-2	UE (Londres)
eu-west-3	UE (París)
ap-northeast-1	Asia Pacífico (Tokio)
ap-northeast-2	Asia Pacífico (Seúl)
ap-northeast-3	Asia Pacífico (Osaka-local)
ap-southeast-1	Asia Pacífico (Singapur)
ap-southeast-2	Asia Pacífico (Sídney)
ap-south-1	Asia Pacífico (Mumbai)
sa-east-1	América del Sur (São Paulo)

Il·lustració 8.5 Regió d'AWS

Ara amb tot instal·lat, podem configurar Corosync i Pacemaker.

8.2 Configuració de l'alta disponibilitat

El primer pas és crear el clúster que volem monitorar amb Corosync, però com la nostra xarxa no permet multicast abans hem de configurar-ho. Dintre del fitxer /etc/corosync/corosync.conf hem d'introduir la configuració descrita a l'annex A.2.

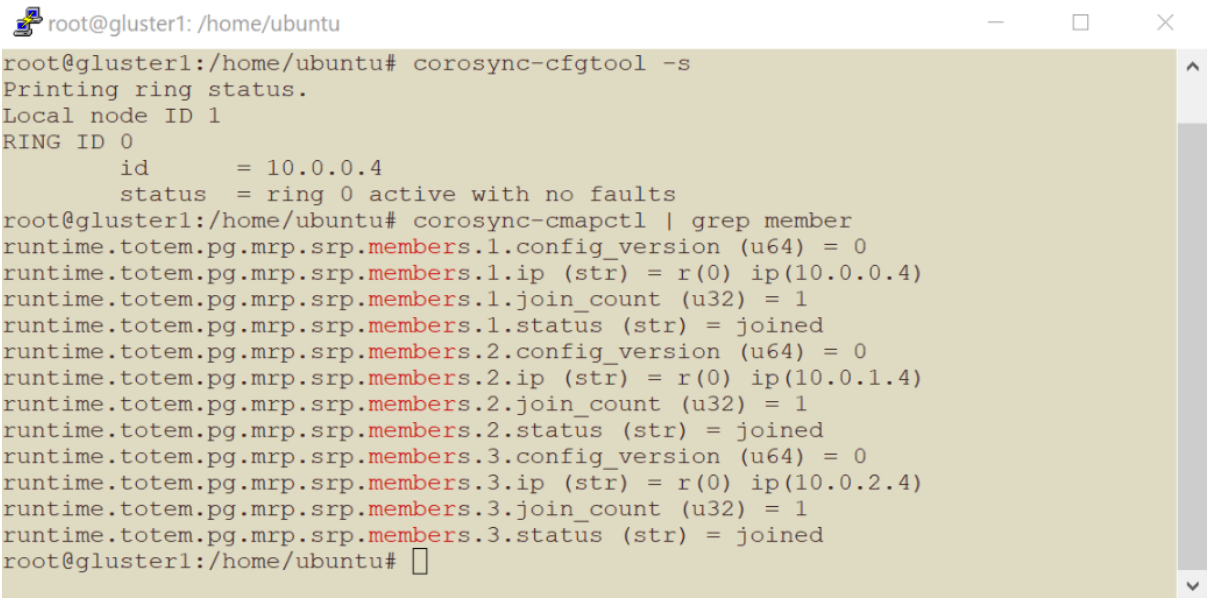
El fitxer ha de ser el mateix a tots tres servidors, amb l'única diferència que la línia *bindnetaddr:* ha de contenir l'adreça IP local de cada servidor, en cas de gluster1 10.0.0.4, per exemple.

Una vegada configurat el fitxer en tots tres servidors, hem d'iniciar el servei Corosync, ho podem fer amb la comanda:

```
> service corosync start
```

Una vegada executat a tots tres servidors, podem executar a qualsevol dels tres les comandes:

```
> corosync-cfgtool -s
> corosync-cmapctl | grep membre
```



Il·lustració 8.6 Estat de Corosync

Si observem la il·lustració 8.6 podem veure que el clúster té tres membres per que identifiquem 3 members.X amb 3 adreces IP i 3 estats, entre d'altres (al resultat de la segona comanda).

Altrament, la primera comanda ens retorna la ID del servidor dintre del clúster (descrit com RING), i l'estatus. Per tant podríem entendre com la segona comanda es una ampliació de la primera sobre tots els servidors i no només local.

Finalment hem de configurar Pacemaker. El primer pas és posar en marxa el servei:

```
> service pacemaker start
```

I, una vegada en marxa, hem d'autentificar els tres servidors amb algun usuari perquè Pacemaker pugui operar dintre d'ells. Amb la instal·lació de Pacemaker es crea un usuari anomenat "hacluster", podem canviar la contrasenya d'accés amb la comanda *passwd*, i així poder utilitzar-ho. Tots tres servidors han de tenir la mateixa contrasenya en aquest usuari. Finalment, amb aquest usuari llest, autentiquem els servidors amb la comanda:

```
> pcs cluster auth 10.0.0.4 10.0.1.4 10.0.2.4
```

La comanda ens demanarà l'usuari i la contrasenya, introduirem els que hem preparat al pas anterior (hacluster). És recomanable posar les adreces IP i no els noms en aquesta comanda per què no funciona correctament sinó.

Com ja sabem, Pacemaker monitora recursos, en venen molts per defecte instal·lats, però cap pensat per al nostre servei, per tant n'hem d'introduir un personalitzat. Aquest servei ha de comprovar que l'adreça IP flotant funcioni correctament, i en cas contrari enviar un missatge a AWS perquè s'assigni a un dels altres components del clúster.

Per crear un recurs personalitzat, hem de crear un fitxer amb permisos 755 a la ruta /usr/lib/ocf/resource.d/heartbeat/, el nom d'aquest fitxer serà el que s'utilitzi per crear el recurs, en el nostre cas utilitzarem el nom *awseip*. A l'annex A.3 podem veure el codi del fitxer.

Amb el recurs creat per monitorar una adreça, executem la següent comanda:

```
> pcs resource create elasticAddr ocf:heartbeat:awseip /
 elastic_ip="<elastic-ip>" awscli="$(which aws)" /
 allocation_id="<allocation-id>" /
 op start   timeout="60s" interval="0s"  on-fail="stop" /
 op monitor timeout="20s" interval="5s" on-fail="restart" /
 op stop    timeout="60s" interval="0s"  on-fail="block"
```

En aquesta comanda especifiquem cada quant volem que és monitori l'adreça IP, i hem d'escriure la IP Flotant (Elastic IP) i l'Allocation ID que hem vist al pas de crear la IP Elàstica.

Si volem comprovar si tot està correctament configurat, podem executar la comanda:

```
> pcs resource show
```

Que ens mostrarà els actuals recursos, o:

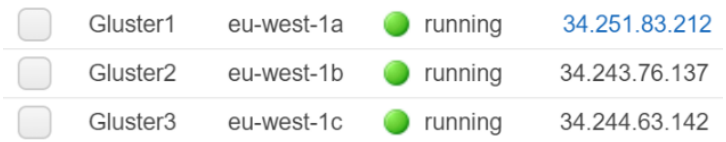
```
> pcs status
```

Que ens mostrarà l'estat general de Pacemaker.

8.3 Validació de l'alta disponibilitat

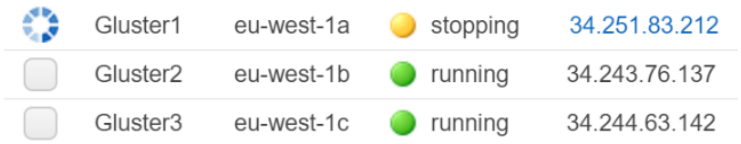
Com a prova, apagarem manualment un dels servidors i utilitzant la consola web d'AWS observarem el comportament dels servidors, i monitorar que ocorre amb les adreces IP per veure si, efectivament, aquestes són dinàmiques.

L'estat inicial consisteix en, tots tres servidors funcionant, per defecte el primer d'ells agafa la IP elàstica i se l'assigna. Els altres dos servidors tenen adreces IP temporals diverses. Podem observar a la il·lustració 8.7 com l'adreça IP flotant és remarcada en blau.



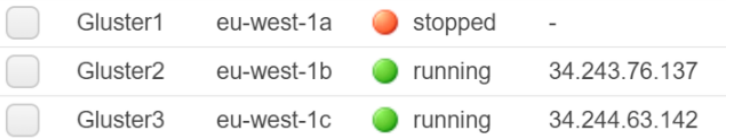
Il·lustració 8.7 Estat inicial

A continuació apaguem la màquina que té l'adreça flotant (il·lustració 8.8):



Il·lustració 8.8 Gluster1 apagant-se

I observem com l'adreça IP es desassigna del servidor a la il·lustració 8.9:



Il·lustració 8.9 Elastic IP alliberada

Ara, si refresquem la consola, veiem a la il·lustració 8.10 com la IP s'ha canviat.

Il·lustració 8.10 Elastic IP assignada

Efectivament la nostra adreça IP Elàstica torna a estar assignada, ara el servidor 2 és qui rebrà les connexions. Per tant, analitzem la situació:

- AWS ofereix una disponibilitat mensual de 99.99% al seu servei EC2 (disponible al seu SLA [14]).

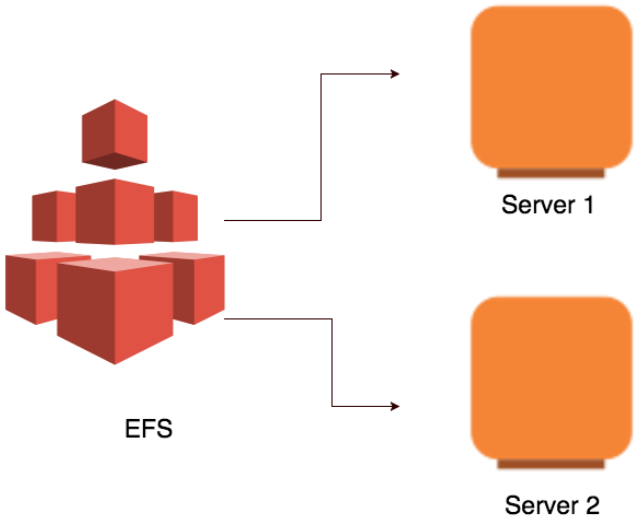
- Amb el 99.99%, en un mes tenim de mitja 4,32 minuts de caiguda (utilitzant el servei ininterrompudament).

- Combinant els tres servidors (perquè, com podem observar, estan allotjats a 3 zones diferents), obtenim una disponibilitat de 99.999999% (la possibilitat que tots tres deixin de funcionar és del 0.000001%). Vol dir que de mitja al mes passem de 4,32 minuts a 0,0432 minuts, o sigui 2,59 segons.

Per tant ara tenim una disponibilitat molt millorada, juntament amb el sistema funcionant.

9. EFS

EFS és la solució d'AWS per oferir un disc en xarxa. A diferència de GlusterFS, es totalment mantingut per AWS, el client només ha de connectar-se i pot començar a utilitzar el disc sense cap desplegament de cap màquina ni res semblant (Ara però veurem com en el cas de Windows haurem de canviar un detall abans de poder fer-ho servir). A la il·lustració 9.1 podem veure la simplicitat del disseny.



Il·lustració 9.1 Funcionament d'EFS[4]

Per comparar el nostre sistema amb GlusterFS, crearem un disc remot a AWS amb EFS, i analitzarem el rendiment. Per tant enfrontarem EFS amb GlusterFS.

EFS per defecte només funciona amb NFS versió 4. Això ens presenta un problema doncs com hem comentat a l'apartat de GlusterFS, Windows només suporta versió 3 nativament. Nosaltres utilitzarem un client desenvolupat a la universitat de Michigan [15] per fer servir NFS Versió 4.

Però encara i així, aquest client ens presenta un problema, el servei EFS d'AWS té certes operacions restringides, en concret l'operació d'NFS `OPEN4_SHARE_DENY_NONE.` Però en ser un client de codi lliure, existeix una versió modificada i preparada prèviament per aquest ús concretament [16].

Per tant, hem compilat i executat aquesta versió, funcionant sense cap mena de problema.

[4] infinitypp.com/amazon-aws/scalable-wordpress-architecture-on-aws/

9.1 Creació de l'EFS

Per crear un EFS, hem de dirigir-nos al servei EFS dintre de la pestanya *Storage* (a AWS), i seleccionar l'opció *Create file system*. Amb la configuració per defecte ens serveix sense cap problema, només hem d'assegurar-nos que el client que utilitzi l'EFS estigui dins de la mateixa xarxa que l'EFS.

Una vegada tenim creat l'EFS cal inicialitzar-ho, d'altra manera no tindrem permisos per escriure. Per inicialitzar el disc, a causa de no estar pensat per Windows, necessitem connectar el disc a algun servidor Linux, i executar una assignació de permisos:

```
> mount -t nfs4 -o
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans
=2 <NOM_DNS_EFS>:/ ./efs
```

D'aquesta manera muntem el disc a la carpeta **efs**, a la ruta actual, el nom DNS del disc EFS el podem consultar a la consola d'AWS, a la mateixa pestanya on hem creat el disc. Una vegada muntat, podem assignar els permisos:

```
> chmod a+rwx ./efs
```

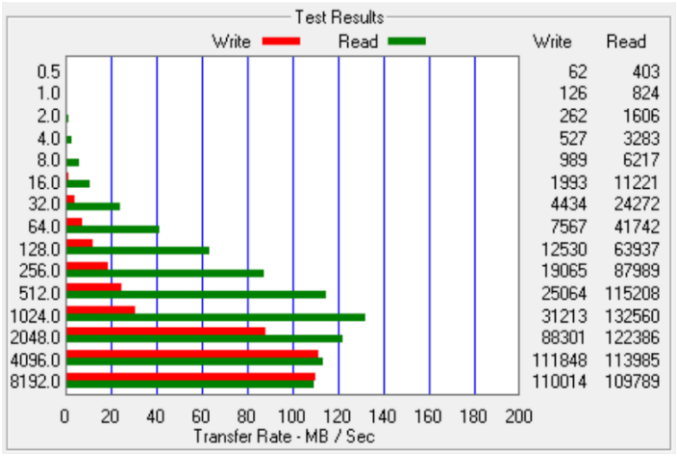
9.2 Connexió a Windows

Ara, amb el disc EFS inicialitzat, podem muntar el disc a Windows i analitzar el rendiment. Utilitzant el client mencionat abans, muntem el disc utilitzant el DNS de l'EFS (podem veure com el procés total ha sigut molt més senzill):

```
> nfs_mount.exe R: <EFS_DNS>:\
```

9.3 Execució de tests i anàlisi de rendiment

Per avaluar el rendiment, al igual que amb GlusterFS, executarem el test de velocitat ATTO, i provarem el temps per executar el videojoc DOOM.



Il·lustració 9.2 Resultat amb EFS

Com podem observar a la il·lustració 9.2, els resultats també són bons, obtenim inclús millors pics de velocitat al test. Per tant per esclarir dubtes, provem de executar el videojoc DOOM.

L'execució del videojoc DOOM fins al primer frame ha tardat 40 segons en el test realitzat (de mitja, ja que s'ha provat vàries vegades per assegurar). Això es 12 segons més lent que el disc dur i quasi el doble que GlusterFS.

Ara bé, durant l'execució d'aplicacions, EFS ha funcionat amb bona velocitat igual que GlusterFS (dintre del que es pot esperar d'un disc en xarxa), però alguna vegada sense motiu aparent, les aplicacions es congelen i la connexió al disc EFS es penja.

Aquest error no sembla seguir un patró concret i s'ha tractat de solucionar sense èxit. És més greu del que sembla, ja que una empresa no pot oferir el seu producte si aleatòriament algun client es quedarà penjat, o si cal reiniciar alguna connexió cada x temps.

10. Conclusions i treball futur

Recordem, l'objectiu d'aquest treball és crear una plataforma al núvol que actuí de sistema de fitxers remot. Hem aconseguit dur-ho a terme de dues formes:

- Utilitzant GlusterFS juntament amb Corosync/Pacemaker per l'alta disponibilitat, sent aquesta la opció que ha passat millor els tests per validar el sistema, executant correctament les aplicacions, amb velocitat propera a un disc físic i mantenint el sistema funcionant en cas de caiguda d'algun servidor.

- Creant un sistema de fitxers remot amb el servei EFS d'AWS, passant igualment els tests del projecte per validar-ho, executant correctament aplicacions, amb una velocitat realment bona, però amb algun problema puntual en la connexió amb els clients.

10.1 Conclusions tècniques

Seguint estrictament les qüestions tècniques:

- EFS es clarament molt més fàcil de desplegar

- GlusterFS ofereix un millor rendiment en general, millorant els temps de latència sobretot. Aquest aspecte és important per al servei d'aplicacions en temps real.

- L'alta disponibilitat en EFS es igual de senzilla que el desplegament, de tot s'encarrega AWS. Ara bé, estem limitats a connectar-nos amb màquines d'AWS i per regions.

Per tant, EFS seria ideal per un entorn més petit, potser per utilitzar de forma personal o en empreses petites, i/o que utilitzin majoritàriament Linux. En canvi, encara que GlusterFS sigui més complicat de desplegar i de mantenir, ofereix millor rendiment i no depenen de cap proveïdor directament. Si volem connectar-nos des de qualsevol altra empresa podem tranquil·lament. Així doncs com a decisió s'escull GlusterFS.

10.2 Conclusions personals

Utilitzant GlusterFS tenim d'un sistema més complex en nivell d'ús i tècnic, ja que requereix tot el procés de muntatge, escollir hardware adient per al nostre servei, configurar la xarxa... Una vegada preparat GlusterFS, hem de configurar l'alta disponibilitat (que en el nostre cas ha sigut bastant complicat en haver de lidiar amb una xarxa amb el multicast restringit), etcètera.

En canvi amb EFS pràcticament és un parell de clics posar-ho en marxa, però per contra no té cap client natiu ni oficial per Windows (amb els problemes que ha comportat això), a més d'un punt molt important, es exclusiu d'AWS, si canviem de cloud no podem fer-ho servir (o amb més costos de transferència i pitjor rendiment).

10.3 Treball Futur

El treball s'ha pogut dur a terme satisfactòriament, una plataforma ha pogut ser desenvolupada, provada i comparada amb altres solucions. La solució existeix, ara bé, un dels punts millorables del sistema és el fet de connectar-nos als servidors manualment. S'hauria d'implementar un balancejador de càrrega, d'aquesta forma podríem repartir les connexions amb els N servidors que conformin el clúster de manera dinàmica (i automàtica) aprofitant encara més la velocitat màxima de tot el clúster. Alhora que monitorar si un servidor cau i no enviar cap client a aquest.

Com a altra opció de treball futur també s'hauria de provar amb un gran nombre d'accessos, per què no és el mateix 50, que 500 ni 5000 clients connectats al sistema de fitxers remot. S'hauria de dimensionar el sistema correctament segons les necessitats que tingui LudiumLab.

11. Bibliografia

Referències

[1] What is GlusterFS? (sense data). Consultat el 4 / Març / 2018, a http://docs.gluster.org/en/latest/Administrator%20Guide/GlusterFS%20Introduction/

[2] HDFS Architecture. (sense data). Consultat el 4 / Març / 2018, a https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[3] What is Mapreduce. (sense data). Consultat el 4 / Març / 2018, a https://www.ibm.com/analytics/hadoop/mapreduce

[4] Institutions powered by Hadoop. (sense data). Consultat el 4 / Març / 2018, a https://wiki.apache.org/hadoop/PoweredBy

[5] Ceph storage. (sense data). Consultat el 4 / Març / 2018, a https://ceph.com/ceph-storage/

[6] Object oriented storage. (sense data). Consultat el 5 / Març / 2018, a https://www.scality.com/blog/object-storage-will-help-find-car/

[7] AWS EFS. (sense data). Consultat el 4 / Març / 2018, a https://aws.amazon.com/es/efs/

[8] AWS EFS Performance. (sense data). Consultat el 4 / Març / 2018, a https://docs.aws.amazon.com/efs/latest/ug/performance.html

[9] Metodologia Scrum. (sense data). Consultat el 5 / Març / 2018, a https://proyectosagiles.org/que-es-scrum/

[10] Jira software. (sense data). Consultat el 4 / Març / 2018, a https://es.atlassian.com/software/jira

[11] Atlassian software. (sense data). Consultat el 4 / Març / 2018, a https://es.atlassian.com/software/confluence

[12] Fiabilidad de las marcas de portátiles. (sense data). Consultat el 17 / Març / 2018, a http://tecnicosclic.com/blog/que-portatil-no-te-recomendariamos-comprar-cual-es-mas-fiable/

[13] Que es fstab. (sense data). Consultat el 9 / Juny / 2018, a https://wiki.archlinux.org/index.php/Fstab_(Espa%C3%B1ol)

[14] AWS EC2 Contrato de nivel de servicio. (sense data). Consultat el 9 / Juny / 2018, a https://aws.amazon.com/es/ec2/sla/

[15] NFS Client v4 Michigan University. (sense data). Consultat el 10 / Juny / 2018, a http://www.citi.umich.edu/projects/nfsv4/windows/readme.html

[16] client NFS v4 Modificat. (sense data). Consultat el 10 / Juny / 2018, a https://github.com/kofemann/ms-nfs41-client

[17] Gluster Architecture. (sense data). Consultat el 4 / Juny / 2018, a https://docs.gluster.org/en/v3/Quick-Start-Guide/Architecture/

[18] Red Hat Gluster Requirements. (sense data). Consultat el 4 / Juny / 2018, a https://access.redhat.com/articles/66206
[19] NFS vs CIFS performance. (sense data). Consultat el 4 / Juny / 2018, a https://www.anandtech.com/show/7071/synology-ds1812-8bay-smb-soho-nas-review/4
[20] What is Corosync. (sense data). Consultat el 8 / Juny / 2018, a https://en.wikipedia.org/wiki/Corosync_Cluster_Engine
[21] Pacemaker Clusterlabs. (sense data). Consultat el 8 / Juny / 2018, a https://wiki.clusterlabs.org/wiki/Pacemaker
[22] Putty.org. (sense data). Recollit de https://putty.org/
[23] TightVNC. (sense data). Recollit de https://www.tightvnc.com/
[24] ATTO Benchmark. (sense data). Recollit de https://www.atto.com/disk-benchmark/
[25] GlusterFS. (sense data). Recollit de https://docs.gluster.org/en/v3/Administrator%20Guide/GlusterFS%20Introduction/
[26] Team Gantt (sense data). Recollit de https://app.teamgantt.com
[27] Google Drive. (sense data). Recollit de https://drive.google.com

Annex A

A.1 Política d'AWS

Política de seguretat aplicable a un perfil d'AWS per tal de permetre moure, assignar i desassignar adreces IP d'AWS, tant Elàstiques com privades. En el nostre cas s'utilitza per crear un perfil i automatitzar l'alta disponibilitat.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DisassociateAddress",
                "ec2:DescribeAddresses",
                "ec2:DescribeRegions",
                "ec2:DescribeInstances",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:AssignPrivateIpAddresses",
                "ec2:AssociateAddress"
            ],
            "Resource": "*"
        }
    ]
}
```

A.2 Codi de /etc/corosync/corosync.conf

Configuració provada i funcional per a crear un clúster amb Corosync i treballar en una xarxa sense multicast, com per exemple les xarxes d'AWS.

```
totem {
        version: 2

        # Corosync itself works without a cluster name, but DLM needs one.
        # The cluster name is also written into the VG metadata of newly
        # created shared LVM volume groups, if lvmlockd uses DLM locking.
        # It is also used for computing mcastaddr, unless overridden below.
        cluster_name: gluster

        # How long before declaring a token lost (ms)
        token: 3000
```

```
  # How many token retransmits before forming a new configuration
  token_retransmits_before_loss_const: 10

  # Limit generated nodeids to 31-bits (positive signed integers)
  clear_node_high_bit: yes

  # crypto_cipher and crypto_hash: Used for mutual node authentication.
  # If you choose to enable this, then do remember to create a shared
  # secret with "corosync-keygen".
  # enabling crypto_cipher, requires also enabling of crypto_hash.
  # crypto_cipher and crypto_hash should be used instead of deprecated
  # secauth parameter.

  # Valid values for crypto_cipher are none (no encryption), aes256,
aes192, aes128 and  3des. Enabling crypto_cipher, requires also
enabling of crypto_hash.
  crypto_cipher: none

  # Valid values for crypto_hash are none md5, sha1, sha256, sha384 and
sha512.
  crypto_hash: none

  # Optionally assign a fixed node id (integer)
  # nodeid: 1234

  transport: udpu #IMPORTANT, BECAUSE AWS DOESN'T SUPPORT MULTICAST

  # interface: define at least one interface to communicate
  # over. If you define more than one interface stanza, you must
  # also set rrp_mode.
  interface {
          # Rings must be consecutively numbered, starting at 0.
          ringnumber: 0
          # This is normally the *network* address of the
          # interface to bind to. This ensures that you can use
          # identical instances of this configuration file
          # across all your cluster nodes, without having to
          # modify this option.
          bindnetaddr: <LOCAL-IP> #SERVER IP ADDRESS
          # However, if you have multiple physical network
          # interfaces configured for the same subnet, then the
          # network address alone is not sufficient to identify
          # the interface Corosync should bind to. In that case,
          # configure the *host* address of the interface
```

```
                # instead:
                # bindnetaddr: 192.168.1.1
                # When selecting a multicast address, consider RFC
                # 2365 (which, among other things, specifies that
                # 239.255.x.x addresses are left to the discretion of
                # the network administrator). Do not reuse multicast
                # addresses across multiple Corosync clusters sharing
                # the same network.
                # mcastaddr: 239.255.1.1
                # Corosync uses the port you specify here for UDP
                # messaging, and also the immediately preceding
                # port. Thus if you set this to 5405, Corosync sends
                # messages over UDP ports 5405 and 5404.
                mcastport: 5405
                # Time-to-live for cluster communication packets. The
                # number of hops (routers) that this ring will allow
                # itself to pass. Note that multicast routing must be
                # specifically enabled on most network routers.
                ttl: 1
        }
}

logging {
        # Log the source file and line where messages are being
        # generated. When in doubt, leave off. Potentially useful for
        # debugging.
        fileline: off
        # Log to standard error. When in doubt, set to no. Useful when
        # running in the foreground (when invoking "corosync -f")
        to_stderr: no
        # Log to a log file. When set to "no", the "logfile" option
        # must not be set.
        to_logfile: no
        #logfile: /var/log/corosync/corosync.log
        # Log to the system log daemon. When in doubt, set to yes.
        to_syslog: yes
        # Log with syslog facility daemon.
        syslog_facility: daemon
        # Log debug messages (very verbose). When in doubt, leave off.
        debug: off
        # Log messages with time stamps. When in doubt, set to on
        # (unless you are only logging to syslog, where double
        # timestamps can be annoying).
        timestamp: on
        logger_subsys {
```

```
                subsys: QUORUM
                debug: off
        }
}

quorum {
        # Enable and configure quorum subsystem (default: off)
        # see also corosync.conf.5 and votequorum.5
        provider: corosync_votequorum
        expected_votes: 2
}

nodelist {
  node {
    ring0_addr: 10.0.0.4
    name: gluster1
    nodeid: 1
  }
  node {
    ring0_addr: 10.0.1.4
    name: gluster2
    nodeid: 2
  }
  node {
    ring0_addr: 10.0.2.4
    name: gluster3
    nodeid: 3
  }
}
```

A.3 Pacemaker awseip resource

Aquest codi permet crear un recurs personalitzat per monitorar una IP Elàstica d'AWS i emular el comportament d'una adreça IP flotant. Treballa conjuntament amb la consola d'AWS, per tant cal tenir un compte prèviament configurat com veiem a l'apartat 8.

```
#!/bino/sh
#
#
#    Manage Elastic IP with Pacemaker
#
#
# Copyright 2016 guessi <guessi@gmail.com>
#
# Licensed under the Apache License, Version 2.0 (the "License");
```

```
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
#

#
#  Prerequisites:
#
#   - preconfigured AWS CLI running environment (AccessKey, SecretAccessKey,
etc.                                                                    )
#  - a reserved secondary private IP address for EC2 instances high availablity
#  - IAM user role with the following permissions:
#    * DescribeInstances
#    * AssociateAddress
#    * DisassociateAddress
#

#######################################################################
# Initialization:
OCF_ROOT=/usr/lib/ocf
: ${OCF_FUNCTIONS_DIR=${OCF_ROOT}/resource.d/heartbeat}
. ${OCF_FUNCTIONS_DIR}/.ocf-shellfuncs

#######################################################################

#
# Defaults
#
OCF_RESKEY_awscli_default="/usr/bin/aws"
OCF_RESKEY_profile_default="default"
OCF_RESKEY_api_delay_default="3"

: ${OCF_RESKEY_awscli=${OCF_RESKEY_awscli_default}}
: ${OCF_RESKEY_profile=${OCF_RESKEY_profile_default}}
: ${OCF_RESKEY_api_delay=${OCF_RESKEY_api_delay_default}}

meta_data() {
    cat <<END
<?xml version="1.0"?>
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
<resource-agent name="awseip">
<version>1.0</version>
```

```
<longdesc lang="en">
Resource Agent for Amazon AWS Elastic IP Addresses.

It manages AWS Elastic IP Addresses with awscli.

Credentials needs to be setup by running "aws configure".

See https://aws.amazon.com/cli/ for more information about awscli.
</longdesc>
<shortdesc lang="en">Amazon AWS Elastic IP Address Resource Agent</shortdesc>

<parameters>

<parameter name="awscli" unique="0">
<longdesc lang="en">
command line tools for aws services
</longdesc>
<shortdesc lang="en">aws cli tools</shortdesc>
<content type="string" default="${OCF_RESKEY_awscli_default}" />
</parameter>

<parameter name="profile">
<longdesc lang="en">
Valid AWS CLI profile name (see ~/.aws/config and 'aws configure')
</longdesc>
<shortdesc lang="en">profile name</shortdesc>
<content type="string" default="${OCF_RESKEY_profile_default}" />
</parameter>

<parameter name="elastic_ip" unique="1" required="1">
<longdesc lang="en">
reserved elastic ip for ec2 instance
</longdesc>
<shortdesc lang="en">reserved elastic ip for ec2 instance</shortdesc>
<content type="string" default="" />
</parameter>

<parameter name="allocation_id" unique="1" required="1">
<longdesc lang="en">
reserved allocation id for ec2 instance
</longdesc>
<shortdesc lang="en">reserved allocation id for ec2 instance</shortdesc>
<content type="string" default="" />
</parameter>

<parameter name="private_ip_address" unique="1" required="0">
<longdesc lang="en">
predefined private ip address for ec2 instance
</longdesc>
<shortdesc lang="en">predefined private ip address for ec2 instance</shortdesc>
<content type="string" default="" />
```

```
</parameter>

<parameter name="api_delay" unique="0">
<longdesc lang="en">
a short delay between API calls, to avoid sending API too quick
</longdesc>
<shortdesc lang="en">a short delay between API calls</shortdesc>
<content type="integer" default="${OCF_RESKEY_api_delay_default}" />
</parameter>

</parameters>

<actions>
<action name="start"        timeout="30s" />
<action name="stop"         timeout="30s" />
<action name="monitor"      timeout="30s" interval="20s" depth="0" />
<action name="migrate_to"   timeout="30s" />
<action name="migrate_from" timeout="30s" />
<action name="meta-data"    timeout="5s" />
<action name="validate"     timeout="10s" />
<action name="validate-all" timeout="10s" />
</actions>
</resource-agent>
END
}

#######################################################################

awseip_usage() {
    cat <<END
usage:           $0 {start|stop|monitor|migrate_to|migrate_from|validate|validate-
all|meta                                                              -data}

Expects to have a fully populated OCF RA-compliant environment set.
END
}

awseip_start() {
    awseip_monitor && return $OCF_SUCCESS

    if [ -n "${PRIVATE_IP_ADDRESS}" ]; then
        NETWORK_INTERFACES_MACS="$(curl     -s http://169.254.169.254/latest/meta-
data/network/interfaces/macs/)"
        for MAC in ${NETWORK_INTERFACES_MACS}; do
            curl              -s              http://169.254.169.254/latest/meta-
data/network/interfaces/macs/${MAC}/local-ipv4s |
                grep -q "^${PRIVATE_IP_ADDRESS}$"
            if [ $? -eq 0 ]; then
                NETWORK_ID="$(curl          -s http://169.254.169.254/latest/meta-
data/network/interfaces/macs/${MAC}/interface-id)"
            fi
```

```
        done
        $AWSCLI --profile $OCF_RESKEY_profile ec2 associate-address  \
            --network-interface-id ${NETWORK_ID} \
            --allocation-id ${ALLOCATION_ID} \
            --private-ip-address ${PRIVATE_IP_ADDRESS}
        RET=$?
    else
        $AWSCLI --profile $OCF_RESKEY_profile ec2 associate-address  \
            --instance-id ${INSTANCE_ID} \
            --allocation-id ${ALLOCATION_ID}
        RET=$?
    fi

    # delay to avoid sending request too fast
    sleep ${OCF_RESKEY_api_delay}

    if [ $RET -ne 0 ]; then
        return $OCF_NOT_RUNNING
    fi

    ocf_log info "elastic_ip has been successfully brought up (${ELASTIC_IP})"
    return $OCF_SUCCESS
}

awseip_stop() {
    awseip_monitor || return $OCF_SUCCESS

    ASSOCIATION_ID=$($AWSCLI  --profile  $OCF_RESKEY_profile  --output  json  ec2
des                                                              cribe-addresses \
                        --allocation-id     ${ALLOCATION_ID}     |     grep     -
m 1 "Associati                                                       onId" |
awk -F'"' '{print$4}')
    $AWSCLI --profile $OCF_RESKEY_profile ec2 disassociate-address  \
        --association-id ${ASSOCIATION_ID}
    RET=$?

    # delay to avoid sending request too fast
    sleep ${OCF_RESKEY_api_delay}

    if [ $RET -ne 0 ]; then
        return $OCF_NOT_RUNNING
    fi

    ocf_log info "elastic_ip has been successfully brought down (${ELASTIC_IP})"
    return $OCF_SUCCESS
}

awseip_monitor() {
    $AWSCLI  --profile  $OCF_RESKEY_profile  ec2  describe-instances  --instance-
id "                                                             ${INSTANCE_ID}" |
grep -q "${ELASTIC_IP}"
```

```
    RET=$?

    if [ $RET -ne 0 ]; then
        return $OCF_NOT_RUNNING
    fi
    return $OCF_SUCCESS
}

awseip_validate() {
    check_binary ${AWSCLI}

    if [ -z "$OCF_RESKEY_profile" ]; then
        ocf_exit_reason "profile parameter not set"
        return $OCF_ERR_CONFIGURED
    fi

    if [ -z "${INSTANCE_ID}" ]; then
        ocf_exit_reason "instance_id not found. Is this a EC2 instance?"
        return $OCF_ERR_GENERIC
    fi

    return $OCF_SUCCESS
}

case $__OCF_ACTION in
    meta-data)
        meta_data
        exit $OCF_SUCCESS
        ;;
esac

AWSCLI="${OCF_RESKEY_awscli}"
ELASTIC_IP="${OCF_RESKEY_elastic_ip}"
ALLOCATION_ID="${OCF_RESKEY_allocation_id}"
PRIVATE_IP_ADDRESS="${OCF_RESKEY_private_ip_address}"
INSTANCE_ID="$(curl -s http://169.254.169.254/latest/meta-data/instance-id)"

case $__OCF_ACTION in
    start)
        awseip_validate
        awseip_start
        ;;
    stop)
        awseip_stop
        ;;
    monitor)
        awseip_monitor
        ;;
    migrate_to)
```

```
        ocf_log         info "Migrating         ${OCF_RESOURCE_INSTANCE}         to
${OCF_RESKEY_CRM_met                                                              a
_migrate_target}."
        awseip_stop
        ;;
    migrate_from)
        ocf_log         info "Migrating         ${OCF_RESOURCE_INSTANCE}         from
${OCF_RESKEY_CRM_m                                                                eta
_migrate_source}."
        awseip_start
        ;;
    reload)
        ocf_log info "Reloading ${OCF_RESOURCE_INSTANCE} ..."
        ;;
    validate|validate-all)
        awseip_validate
        ;;
    usage|help)
        awseip_usage
        exit $OCF_SUCCESS
        ;;
    *)
        awseip_usage
        exit $OCF_ERR_UNIMPLEMENTED
        ;;
esac

rc=$?
ocf_log debug "${OCF_RESOURCE_INSTANCE} $__OCF_ACTION : $rc"
exit $rc
```